Knowledge-based Query Formulation

in Information Retrieval

Knowledge-based Query Formulation *in Information Retrieval*

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit Maastricht,
op gezag van de Rector Magnificus,
Prof. dr. A.C. Nieuwenhuijzen Kruseman,
volgens het besluit van het College van Decanen,
in het openbaar te verdedigen
op donderdag 14 september 2000 om 14:00 uur

door

Rudolf Wolfgang van der Pol

Promotores:

Prof. dr. H.J. van den Herik
Prof. dr. ir. J.L.G. Dietz (Technische Universiteit Delft)
Prof. dr. ir. A. Hasman

Leden van de beoordelingscommissie:

Prof. dr. H. Visser (voorzitter)
Prof. dr. ir. K.L. Boon
Prof. dr. K. Järvelin (Universiteit Tampere, Finland)
Prof. dr. H. Koppelaar (Technische Universiteit Delft)
Prof. dr. A.J. van Zanten

SIKS Dissertation Series No. 2000-5
ISBN 90-801577-4-0
Uitgeverij Phidippides, Cadier en Keer
© 2000 Ruud van der Pol
Cover design and lay-out: Rob Molthoff

Contents

Preface

In 1992 I chose to be a Ph.D. student in Maastricht, in order to learn about human problem solving and the modelling thereof, and also to improve my research skills. I am happy and grateful that I had this opportunity for self-fulfilment, and regard it a privilege that in problem solving I even got more chances to develop new ideas than I could have foreseen.

Even though the research took many moments of solitary labour, in fact it was an utterly social activity. The result would not have existed without a variety of contributions, made by several persons. In this preface I would like to acknowledge the various contributions.

In particular, I would like to thank my three supervisors Jaap van den Herik, for his stimulating and energetic efforts to teach me meticulously scientific writing and thinking, for the many other lessons I learned from him, and also for providing the opportunity to do research. His optimistic and erudite personality will always be an example for me. Jan Dietz I want to thank for his lessons in self-criticism and his patience in modelling issues. Arie Hasman provided inspiring enthusiasm and good advice in experimental issues.

The research reported in this thesis started out as part of a larger project, the ARCHIMEDES project. I thank the "senior" members Jaap Hage, Maarten van der Meulen (and his wife Inge Uljee), Georges Span, and my fellow Ph.D. students of that project: Bart Verheij (also for his assistance in notational issues), Arno Lodder, Floris Wiesman, and Bob Quast for their companionship. I have learned much from the co-operation in the early days of the project, when we were all mere freshmen in the field of research and computer science. I want to emphasise Floris Wiesman's support. He always had an open mind for my ideas (in Information Retrieval and elsewhere) and accompanied my thinking with patience and enthusiasm. He has become a friend rather than a colleague.

Moreover, I thank the test persons (students and alumni of Delft University of Technology) who gave generously of their time and intellect in doing the test tasks. I thank Rob Porsius for his efforts in making relevance judgements of the documents retrieved in the experiments, not to mention his generous friendship. The university library of Delft University of Technology is thanked for providing equipment and housing for the experiments.

During the research I have spent much time on the Department of Computer Science. I have enjoyed the contacts with my colleagues. Especially the discussions with Sabine Vanhouwe, often about Triumphs, Rovers, and Jaguars, cheered me up. The patience, bright remarks, and good care of Joke Hellemons made me feel comfortable, and Jo Beerens' efforts and labour made my dependence on computers quite bearable. I am grateful to Antal van den Bosch and his wife Anne-Marie, for sharing my contemplations on life and science, but before all for their sociability and cheerfulness. My roommate Dennis Breuker revived my interests in maths (especially on the dimensionless number pi) and chess. His lively and witty mind were quite refreshing. My other roommate Jeroen Donkers kept me sharp with animated and intriguing opinions on society and computer science. Eric Postma was always there for fascinating chats on cognitive (neuro-)science, and I enjoyed his pertinent refusal to be serious. Other (former) department members that I would like to thank are: Peter Braspenning, Inge Lemmens, Jacques Lenting, Nico Roos, Patrick Schoo, Ben Sim, Evgueni Smirnov, Jos Uiterwijk, Gerard Vreeswijk, Fred Wan, and Ton Weijters.

Special thanks go to Jaana Kekäläinen, who read earlier drafts of the thesis and gave many useful and precise comments, next to being a friend. I also enjoyed working on Chapter 2 with Henk Visser. His substantiated comments and intense attitude were a source of inspiration and a joy to hang on.

Friends and family are as important as colleagues, yet in a different way. I mention especially Hendrik Jan Brookhuis and Yvonne Verhoeven, Gaby Brouns, Rob Postema, Evert and Lydie Haanappel, Coen Jongsma, Friso Oosterbaan and Theresia van Meeteren, Bas van Putten and Esther Cahen, Daphne Scheiberlich, Pieter Vredevoort and Kirsten van Putten, Erik de Vries and Carole Hinze, and – last yet not least – my mother and father, for their uncompromised belief, care, patience, and the necessary diversion.

Ruud van der Pol Amsterdam, July 2000

Acknowledgements

The research reported in this thesis has been carried out under auspices of SIKS, the Dutch Graduate School for Information and Knowledge Systems. SIKS was acknowledged by the Royal Dutch Academy of Arts and Sciences (KNAW) in 1998.

The research was partly financed by the Foundation for Knowledge-Based Systems (SKBS) as part of the B3.A project. SKBS is a foundation with the goal to improve the level of expertise in The Netherlands in the field of knowledge-based systems and to promote the transfer of knowledge in this field between universities and business companies.

Chapter 1
Searching documents

The thesis reports on research in Information Retrieval, the branch of Computer Science that studies the searching and accessing of documents (Van Rijsbergen, 1979). The thesis focuses on searching.

Searching documents is an actual issue, now that huge quantities of digital documents are available. Many persons try to obtain knowledge by inspecting documents. For this purpose they usually need only a few documents of the many documents offered. Their challenge in searching is to identify, in an efficient manner, precisely the documents they need.

The search is often realised by the execution of a query-based document-search process. In this kind of process, a user expresses the characteristics of the documents he[1] wants in a query. For several reasons, the existing query-based document-search processes have an unsatisfactory effectiveness. The formulation of ineffective queries, i.e., of queries that do not express the characteristics of the documents searched for, is one reason for this.

The thesis investigates how query-based document-search processes may be improved. In particular, it treats the question how knowledge can be used for the formulation of effective queries. This chapter introduces Information Retrieval. Section 1.1 focuses on query-based document-search processes. Section 1.2 motivates the research.

1.1 The central theme: document-search processes

There are two known ways of searching documents in a large collection: (1) applying query-based document-search processes, and (2) browsing (Parsaye *et al.*, 1989).

By applying query-based document-search processes, the user makes a description of the documents he is looking for. That description is then compared to a description of the characteristics of the documents in the collection (normally the comparison is by the contents

1 For the sake of brevity we will use in this thesis only the male gender of nouns and pronouns in all cases where the person referred to could be either a male or a female.

of the documents). It is common practice that in query-based document-search processes a computer does the hard labour of comparing descriptions. In this manner the user is enabled to identify the subcollection of relevant documents with little effort, under the conditions that he generates a query that appropriately describes the information he needs and that the search process is perfect.

When browsing, the documents of the collection, or their descriptions, are inspected instance by instance. The user does not have to describe the information in advance, but he may just browse until he encounters and recognises a document he needs. He thus may easily gather relevant documents. Unfortunately, it is very laborious to identify *all* relevant documents of a large collection: all documents then must be inspected.

Both querying and browsing can be useful, and in their present forms may complement each other. In particular, querying is well suited for directed search, and browsing is well suited for exploratory search. Therefore, it is a mere matter of choice that querying is the focus of this thesis, with browsing playing a minor role.

In the following subsections, we elaborate on query-based document-search processes. Subsection 1.1.1 introduces query-based document-search processes, by outlining them. Subsection 1.1.2 introduces well-known performance measures for these processes.

1.1.1 Concepts of query-based document-search processes

Query-based document-search processes serve the goal of efficient and effective identification of specific documents in large collections of documents. They are based on the idea that a document-search process can, for a large part, be performed as a symbol manipulating process. The symbol manipulating part can be left to computer systems, thereby drastically reducing the amount of human effort required in search processes. As a result, both the efficiency and the effectiveness increase.

Figure 1.1 presents a schematic overview of query-based document-search processes, showing four sub-processes, viz. (1) describing, (2) formulating, (3) matching, and (4) inspecting. The inputs and the outputs of these processes are also shown. The figure is a slightly extended version of the schematic overviews given by Salton and McGill (1983) and by Croft (1993).

The notational convention of the figure is as follows. A rounded box depicts a process, being an instance of the process type mentioned in the box. By process we refer to either a mental process or a symbol-manipulating process. Of course, a mental process may also be a symbol-manipulating process.

A straight box depicts what is used in or produced by a process. In the case of a mental process, this comprises thoughts of the human being performing the process, or expressions thereof (e.g., a text document). In the case of a symbol-manipulating process, the process uses or produces strings.

A bold arrow between two processes indicates that the start of the first process has to precede the start of the second process[2]. A normal arrow originating in a straight box and finishing in a rounded box means that the content mentioned in the straight box is *used* by the process; a

2 We note that, in general, it is not necessary that the preceding process has finished before the next process starts.

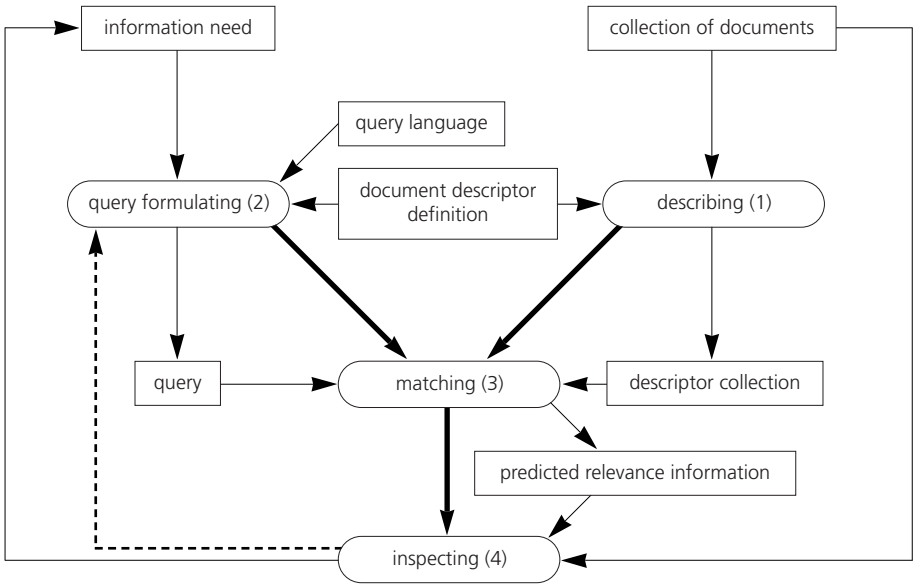

FIGURE 1.1: *Schematic overview of query-based document-search processes.*

plain arrow originating in a rounded box and finishing in a straight box means that the process *produces* the content mentioned in the straight box.

Below, we briefly describe the four sub-processes, which constitute a document-search process. We start with an instance of the *describing* sub-process, drawn at the right side of Figure 1.1 (see (1)). A collection of documents (in the top-right box) is described, i.e., each document is assigned symbols of a descriptor vocabulary: *descriptors*. Descriptors may describe all kinds of characteristics of the document (see Section 3.2), for example title and author. Hardly without an exception, the describing process is performed long before and separately from the other sub-processes. Once a document is assigned descriptors, it is ready for use in search tasks.

Usually, the descriptors are selected with the intention to indicate the subject of the document content[3]; we call such descriptors *indices*. An instance of the description process yielding indexes is called an *indexing process*. Such a process yields a collection of sentences of the type "The document with the identification ... has the index ...". For instance, "The document with the identification d33 has the index chess".

A widely applied type of indexing is *full-text indexing*. Here, all words of a document are used as index terms; barring words which hardly discriminate (such as prepositions). Full-text indexing is easily formalised and executed on a computer. However, it has the disadvantage that not all index terms (even when leaving out the prepositions and the like) appear characteristic

3 It is not necessarily the content that is characterised; see Subsection 1.4.1.

for the content of a document. Other kinds of indexing, not suffering from such a disadvantage, have the drawback of being more difficult to formalise and automate, e.g., conceptual indexing. It has index terms indicating domain concepts. In general, it is difficult to formalise the mapping of the text to terms indicating the characteristic domain concepts. The mapping requires natural language processing and is a subject of intensive and yet slowly progressing research (see, for instance, Evans *et al.,* 1991; Sanderson, 1994).

The second sub-process is the process of *formulating* (see (2)). This process is performed by a user, who assumingly needs information (see the top-left side of Figure 1.1). In other words, he has an *information need* (Croft, 1993). The user formulates (see (2)) his information need in a *query*, being an expression in a formal language that the search system is able to process. For an effective search process (i.e., for meeting the user's information need as far as the document collection and the chosen descriptor type allow for), the query should adequately express the information need.

As a condition for being effective, a query language must include the descriptors used in the describing sub-process. For this purpose, all query languages offer *keys*. A key is an opening, in which a descriptor, or a part thereof, can be specified. The predominant query languages are *Boolean query languages* (Salton, 1989; Dong and Su, 1997). Queries expressed in languages of this kind have operators and keys as their elements. The basic operators are the conjunction, the disjunction, and the negation. They enable the combination of keys. For instance, if a person needs information about Peltier elements or Peltier devices, he may formulate a Boolean query "Peltier AND (element OR device)".

The third sub-process, *matching* (see (3)), compares a given query with the document descriptors produced by the description process. The process is defined as a string-comparison process, yielding a relevance value for each document. The relevance value indicates to what degree the descriptors of the document (in some metric) match the requirements of the query. The relevance value serves as a prediction of the relevance of the document to the user's information need. In many matching processes the relevance value is binary; it thus divides the document collection into documents presumed relevant and irrelevant, respectively. We call the documents presumed relevant *matched documents*.

With binary relevance values, the output of the matching process only contains the document identifiers of the matched documents. In some matching processes the relevance values are ordered; the order is called a *relevance ranking*. With a relevance ranking, the output of the matching process usually contains a list of pairs of (identifier, relevance value) of the documents, typically only of a pre-defined number of documents presumed sufficiently relevant. In analogy to matching processes yielding binary relevance values, we call the documents presumed sufficiently relevant again *matched documents*.

The fourth sub-process, *inspecting* (see (4)), serves to judge the real relevance of the documents presumed (sufficiently) relevant by the matching process. For this purpose, the documents matched are communicated to the user, by their identifiers, possibly accompanied by other information such as their descriptors. The user then inspects the information presented and may ask for additional information on each document (both descriptors and content; see Subsection 2.5.2).

The inspecting process is usually not shown (as in Salton and McGill, 1983, and in Croft, 1993), in spite of its crucial role in the document-search process. For instance, during the inspecting process, the user's information need may change as a result of his receiving information during the inspection of the documents matched. In Chapter 4, it will be shown that this change plays an important role in the document-search process.

Once the inspecting process has been finished, the user may decide to start a new instance of the formulating process. He may be expected to do so if his information need has changed, or his insights in the proper expression of the information need. In the new instance of the formulating process, he reformulates the original query. Thereupon new instances of the matching (3) and inspecting (4) processes are executed. Thus, the cycle of formulating, matching, and inspecting is repeated. The use of information from a previous cycle for a new cycle is called *relevance feedback*. In Figure 1.1, the possibility of relevance feedback is indicated by the left-most thick arrow (dotted), from the box inspecting (4) to the box formulating (2). The relevance feedback may be repeated several times, until the user is satisfied or expects insufficient benefit from a continuation of the search process (or just wants to stop).

1.1.2 Performance measures

The performance of query-based document-search processes can be measured in terms of efficiency and effectiveness. Efficiency is not elaborated in the thesis. The effectiveness of a search process is usually expressed in terms of its precision and recall. In this thesis, we adopt the latter measures. They are defined as follows (see, e.g., Kent *et al.*, 1955; Swets, 1969; Van Rijsbergen, 1979; Salton and McGill, 1983):

$$precision = \frac{the\ number\ of\ relevant\ documents\ matched}{the\ number\ of\ documents\ matched}$$

$$recall = \frac{the\ number\ of\ relevant\ documents\ matched}{the\ total\ number\ of\ relevant\ documents\ in\ the\ document\ collection}$$

The precision and the recall of a document-search process are mutually dependent: an increase of the recall usually results in a decrease of the precision, and vice versa. However, the relationship is different for each type of document-search process. It therefore is difficult to define a single replacement measure (Salton and McGill, 1983; Hull, 1993). In practice, the precision and the recall together express the effectiveness of a document-search process.

1.2 Motivation of the research

In this section, we elaborate on the three main reasons for our decision of conducting research in knowledge-based query formulation. First, query-based document-search processes have become crucial (Subsection 1.2.1). Second, query-based document-search processes are far from perfect (Subsection 1.2.2). Third, the ARCHIMEDES project provided an opportunity to investigate this challenging research topic (Subsection 1.2.3).

1.2.1 Query-based document search processes have become crucial

Query-based document-search processes have been implemented in computer programs since the 1960s, and were used online since the early 1970s. The programs were mainly present in large organisations, e.g., in university libraries, large public libraries, and private libraries of large companies and government institutions. The first large-scaled program implementing a query-based document-search process was STAIRS, by IBM (Salton and McGill, 1983). It offered full-text indexing and Boolean matching. Although the document descriptors were stored digitally, the documents (books, journals, papers, etc.) were stored in hardcopy format. Nevertheless, by using STAIRS, the laborious task of identifying relevant documents was enlightened. However, accessing those documents remained a laborious job, e.g., consisting of going to a library, copying and/or carrying them to one's room.

Since the early 1980s, documents have been stored increasingly on digital computer systems. This trend is promoted even more by the presence of the World Wide Web. Documents previously published on paper bearers are nowadays converted to a digital format. They are even made available on bearers for mass storage, such as CD-ROM and DVD, or they are made available via communication networks, such as Intranets and the Internet. In this manner, the required physical effort of copying and transporting digital information is small, especially in comparison to the above-mentioned existing ways of information gathering.

The availability of digital documents was made possible by several recent developments in computer technology. These developments have yielded three functionalities of information systems. Below, we briefly discuss the functionalities, viz. (1) the mass storage and the fast access of information, (2) high-speed, high-density data communication, and (3) the handling of multimedia information[4].

First, *the mass storage and the fast access of information* are now available, not only on mainframe computer sites but also on almost every workstation[5]. A major contribution to this functionality is the development of optical discs, most typically exemplified by the CD-ROM, and, recently, the DVD. A DVD may contain up to 17 Gigabyte. The information on DVD can be accessed in a few milliseconds (Nachtmann, 1999). This storage capacity suffices to store over 5 million pages of text, 25 hours of high-quality audio, 25,000 full-colour images or 15 minutes of video, all uncompressed. Compression techniques have increased the storage capacity in the order of magnitude of 100, depending on the quality-loss accepted (Bryan, 1995). Synthetic coding techniques, such as applied in the standard MPEG-4 and the forthcoming MPEG-7, will increase the compression rates even more (Koenen, 1999).

Second, *high-speed, high-density data communication* is now possible between workstations, regardless of their physical distance on earth. This can be ascribed to the arrival of network facilities. Within professional organisations, the work stations are interconnected by (predominantly private) Local Area Networks (LANs); their data communication speeds are in the order of magnitude of 10 to 100 Mbit/s. Among organisations, the local networks are interconnected via interlocal networks, many of which are public networks. Most links in these interlocal net-

4 By multimedia information we refer to a combination of text, photographs, sounds, and movies. The notion is elaborated in Section 3.2.

5 Practically all employees in an office environment dispose of a single-user work station.

works are constituted by satellite communication links and optical fibre links. The maximum speed and density are lower than those for LANs, but steadily increasing. They are already sufficiently high to allow for demanding applications, such as teleconferencing via a video link or the broadcast of video movies[6] (Reid, 1997). The interlocal networks are tied into networks spanning the entire globe. In the Western countries, the networks are already connecting many of the single-user workstations. From the presence of these data-communication facilities, the availability of information is no longer dependent on where the work station or (mainframe) computer site is located.

Third, *the handling of multimedia information* by computers is now common. Earlier, only alphanumeric information could be handled. The multimedia possibilities result from, among other things, increased storage capacity of electronic memory, increased speed of processing units, increased throughput speed of various interfaces, and advanced compression techniques (Hoogeveen, 1995; Ram, 1999). These developments enable the handling of the large data quantities that characterise multimedia information.

The availability of large quantities of digital documents and the easy accessibility (both already mentioned above) have led to a widespread application and an intensive use of query-based document-search processes. On the World Wide Web (WWW) only, tens of such systems are consulted by millions of users daily (Scholtes, 1996; Dong and Su, 1997, Hoogeveen, 1999).

The World Wide Web itself is intensively used for providing information, as an alternative to printed matter. In total, there are well over 300 million home pages on the WWW (Horn, 1999). Though it is still in its childhood, it has already become a major communication means for leisure purposes, and also for doing business (Garner, 1999). Without query-based document-search processes, we could not easily find our way in the 'ocean' of information offered on the WWW, nor in many other large collections of documents. It thus seems reasonable to say that query-based document-search processes are crucial in our lives.

1.2.2 Query-based document-search processes are far from perfect

Although query-based document-search processes are successfully used to simplify document search, their performance is still far from perfect. In the past, their poor performance was convincingly shown in a study by Blair en Maron (1985), at least for the most common kind of document-search processes (having Boolean query languages and full-text indexes). In an experimental study of searching documents, lawyers and paralegals were asked to search until they believed to have a recall of 75%. It appeared that the recall realised had a weighted average of approximately 20%. The precision was 70%. Although the study is dated, the performance problem that it identifies still exists (Blair, 1996). Recently, Jansen *et al.* (1998) have shown that a large part of poor performance in full-text retrieval with Boolean queries can be explained by problems in query formulation. In Chapter 3, we elaborate on query formulation as a cause of poor performance of document-search processes.

---

6 We note that these applications are still under development. Teleconferencing is until now only applied on a small scale, requiring the fastest connections available. The broadcast of video movies still utilises the presence of dedicated routers between sender and receiver, and the quality of the videos is still poor compared to that of television.

The widespread application of query-based document-search processes has made the poor performance a lively topic, also outside the research arena of Information Retrieval. For instance, it is frequently mentioned in magazines and newspapers (see, e.g., Scholtes, 1996; Steketee, 1997; Van Gent and Kraay, 1999). It is our belief that research on query-based document-search processes is a way towards improving such processes.

1.2.3 A research opportunity: the ARCHIMEDES project

A good opportunity for performing the research reported in this thesis was in the ARCHIMEDES project. This project was carried out from 1992 to 1996 at the departments of Computer Science, Medical Informatics, Metajuridica and Management Sciences of Maastricht University. It was part of the SKBS B3-project (1991). The goal of the ARCHIMEDES project was to provide users with a uniform search and access mechanism to multimedia information.

The ARCHIMEDES project has yielded two theses within this scope. The first thesis establishes the basic idea of the project. In that thesis (Wiesman, 1998), search and access are offered by combining four kinds of meta-information in one structure, called the *Information Network*. The meta-information is information about (1) the domain discussed in the documents, (2) the content of the documents (by document descriptors), (3) the relations between the documents, and (4) the locations of the documents. The Information Network can be considered as an extension of a semantic network (for a characterisation of the latter, see Subsection 2.1.5).

The user may search the Information Network using a browsing tool or a (simple) querying tool. The advantage of the new browsing tool over traditional browsing tools is twofold. Primary, browsing is brought to a higher level: the user does not browse mere document content but also browses meta-information. Second, the browsing tool has a graphical user interface (GUI) showing the information in a manner that is clear to the user, and providing an efficient command interface. The browsing tool is called the "ANS browser", in which ANS stands for "ARCHIMEDES Network System".

The second thesis is the present one. It develops the querying tool (named Dipe-D, which means "Description and Identification by propErties, Device"). Dipe-D first supports the user in developing his understanding of the information need, and then formulates a query. In comparison with traditional approaches, this yields a better understanding of the information need, and leads to less errors and mistakes in the formulation process. As a result the document-search process becomes more effective, as will be shown in the thesis.

The thesis also develops a language for knowledge representation (named Dipe-R, which means "Description and Identification by propErties, Representation language"), used by Dipe-D. Dipe-R differs from the Information Network mentioned above. Both the Information Network and Dipe-R have as their basis the idea of a 'network' of statements, each statement being a binary proposition. Dipe-R differs from the Information Network in that it has a different theoretical foundation. Moreover, Dipe-R extends the information network by providing derivation rules. Dipe-R offers a more advanced knowledge representation language with respect to domain knowledge than the Information Network. Since Dipe-R focuses on domain knowledge, it does not have the other three types of information present in the Information Network.

The browsing tool by Wiesman (1998) also plays a role in this thesis: in Section 9.3 we suggest how the browser's GUI and Dipe-D may be integrated, in order to obtain a tool that combines the advantages of both complementary methods.

1.3 Outline of the thesis

The thesis is structured as follows. Chapter 2 provides basic notions essential for document-search processes. Chapter 3 provides an overview of the state of the art in query-based document-search processes, and provides research questions (a problem statement and three questions). The research questions aim at improving the effectiveness of query formulation by using represented knowledge. These questions are elaborated in the Chapters 4 to 8. Chapter 4 deals with the first research question. It provides definitions for concepts essential in query formulation, and it analyses query formulation with respect to process, information, and knowledge. The analysis is based on a case study of query formulation, and yields insights into the process of query formulation. It suggests, among other things, that a fixed procedure can be followed in query formulation. Chapter 5 deals with the second research question, in developing Dipe-R. Chapters 6 to 8 investigate the third research question. Chapter 6 develops Dipe-D, as a formalisation of the fixed procedure suggested in Chapter 3. Chapters 7 and 8 deal with experiments for testing Dipe-D. Chapter 7 describes the implementation of Dipe-D, and the representation in Dipe-R of knowledge in a sample domain. Both implementations are needed in the experiments. Chapter 8 reports on the experiments. The results of the experiments are in support of the tentative conclusion that Dipe-D effectively supports query formulation. Finally, Chapter 9 provides conclusions.

Chapter 2

Basic notions for document-search processes

The notions communication, information, knowledge, and document occur in any description of document-search processes. In this thesis, they are basic notions too. In the previous chapter, we relied on the reader's intuitions of these notions. Unfortunately, such intuitions appear to differ widely among persons. Moreover, in computer science, epistemology, information science, and other fields of study, several definitions have been given for each of these notions. The chapter characterises the four notions, in order to avoid confusion about how we use them, and to contribute to effective communication on query-based document-search processes.

We only provide characterisations, and no definitions, since these notions have arisen as natural concepts, and therefore they have characteristic features but no collection of necessary and sufficient features (as is the case with defined concepts, see Section 5.3). For ease of use, our characterisations are intended to fall within the scope of the wide and often vague common sense notions where possible. However, they are primarily meant to improve the effectiveness and efficiency of communication in our thesis. They may therefore not always fit in common sense notions.

We start our characterisation by providing a context for the characterisations, comprising basic assumptions (Section 2.1). After that, we describe communication (Section 2.2), information (Section 2.3), knowledge (Section 2.4), and document (Section 2.5).

2.1 Basic assumptions

We provide two basic assumptions, concerning a viewpoint with respect to the world (Subsection 2.1.1), and a view on the subjects to which the characterisations apply, i.e., the human beings in this world (Subsection 2.1.2).

2.1.1 The world

We take the results of physics for granted. This means that we work with two main assumptions. The first assumption is that there exists a material world, or universe, in which physical phenomena (e.g., objects and energy) can be distinguished; the phenomena are not necessarily

observable with the bare senses. The second assumption is that there exist regularities in this world which can be explained by physical principles. For example, bodies exert a gravitational force towards each other of which the magnitude is inversely related to their mutual distance.

According to this view the world is not a conception of human beings, i.e., it exists independent of the imagination of human beings. However, it is influenced by the presence of the human beings by their acting. As a consequence, in this world certain events intended by the human beings are possible, and certain events are impossible; human beings are part of this world, and experience in their acting these possibilities and impossibilities. Thus, they may act purposively, i.e., make plans, device scenarios and act accordingly.

For our purpose, it is important that human beings have extended the world by artefacts that are intended to work according to preconceived plans. In our research, these artefacts comprise query-based document-search processes. We strive at a level of theoretical insight that allows us to design document-search processes that are effective in the sense of Subsection 1.1.2.

2.1.2 Human beings

Human beings, and their behaviour, play an essential role in query-based document-search processes. It is therefore necessary to have an understanding of the behaviour of human beings. We take the results of modern psychology for granted, according to which each human being can be regarded a system[7], comprising sensors (e.g., ears, eyes), actuators (e.g., muscles), a signal processor (brains), and a sub-system for transporting signals (essentially comprising the nerve system). Moreover, the human being is an open system, i.e., his behaviour depends both on himself and his environment (Gazzaniga, 1995). The way in which this behaviour takes place is described in this thesis in terms of mental states and processes, and their properties. They comprise thoughts, memories, concepts, goals, meanings, emotions, rationality, consciousness, knowledge, belief, and other mental notions (see, e.g., Ryle, 1949). In other words, we use the traditional vocabulary of psychology. Notwithstanding the usual vagueness of these notions, they are used assuming that the reader is sufficiently acquainted with them in order to dismiss philosophical afterthoughts about them.

It is sometimes emphasised that most mental states and processes seem to arise from something else (i.e., they have an origin), from sensor inputs (e.g., a visual observation made by the person whose mental state or process it is), or from another mental state or process. Audi (1998) recognises the following types of origins: observation including testimony, memory, introspection, reasoning, and inference (Audi, 1998). However, the precise nature of the relationships among subsequent mental states and processes is as yet unknown. In this thesis it suffices to recognise that there is often a kind of causal relationship experienced for subsequent mental states and processes. It is often useful to know the nature of this relationship for assessing the credibility of a mental state or process. For example, you would probably sooner be inclined to believe that there are mass graves in Kosovo from a person who reports that he has seen them with his own eyes than from (possibly manipulated) photographs. For the purpose of assessing credibility of represented mental states and processes, their origin will be represented (in Chapter 5), in terms of the following variables: observation, derivation, communication, and

7 For the notion of system, we refer to, e.g., Bunge (1974).

definition. Hereby, Audi's subdivision is streamlined. At the same time, the central importance of communication processes is recognised. As yet, the analysis of communication processes is best done in terms of the psychological notions introduced above. In the next section, communication processes are looked upon more closely.

2.2 Communication

The subject of communication is discussed here as a background for the subsequent characterisations of the notions of information, knowledge, and documents: information is a product of communication, knowledge can be passed on by communication, and documents are means for communication. The section first elaborates on the mechanism of communication (Subsection 2.2.1). Next, it discusses the relationships among thinking, language (as a means for communication), and the subjects of thinking. The relationships are elaborated in two subsections: one on expressing and interpreting (Subsection 2.2.2), and one on the role of language (Subsection 2.2.3).

2.2.1 The mechanism of communication

In the previous section, we introduced mental states and processes. Human beings interact, i.e., they influence each other by their own acting. Some of these acts (e.g., speech, and gestures) are part of a communication process. Below, we characterise communication processes.

A process of communication is outlined as follows[8]: a person has a mental state or process, and creates on the basis of that mental state or process, or a part thereof, something observable. That something is called a sign. A second person senses the sign and as a result his mental state or process changes. Here, a sign is a physical phenomenon that can be sensed by a person. It may represent a state of affairs in a universe of discourse (i.e., the real world, or a conceived world). Almost anything may serve as a sign, as long as it can be sensed (directly or with the aid of tools). A sign can be an object, e.g., a red dot on a tree (e.g., telling that a hiking trail is there), or a word written on a piece of paper, and it may also be a signal, e.g., a spoken message sent from the mouth of the sender and propagating through the air to the ears of the receiver. Often, a sign is an instance of a sign type (i.e., a symbol); see Subsection 2.2.3.

The sign observed may differ from the sign sent, due to physical events (e.g., distortion of a spoken message during its transport through the air, or a reduced visibility of the red dot on a tree). To facilitate our descriptions of communication processes, some terminology is introduced. The creation of a sign, prompted by a mental state or process given with a certain intention, is called expressing (to be distinguished from expression, which may be the product of the process of expressing). The sensing of a sign and the influencing of (a part of) a mental state or process are called interpreting. For expressing and interpreting, there exist conventions. For communication to be effective, i.e., for the sender to realise the desired effect (a desired change in mental state or process of the receiver) both sender and receiver should be familiar with the con-

8 This outline can be found in any introductory text book on communication science, e.g., Berlo (1960), Akmajian, Demers, and Harnish (1979), Prak (1979), Fiske (1982), and Berger and Chaffee (1987).

ventions. The conventions usually do not occur in isolation, but a part of a collection of conventions in connection with a language. Below, we elaborate on language.

2.2.2 Language

A language can be characterised as a system of signs, submitted to conventions for communication. Following Bunge (1973), we distinguish symbolic and nonsymbolic languages. In symbolic languages, the signs are detached from individual circumstances. The signs are instances of equivalence classes to which the same conventions apply. Such signs are instances of sign types (or symbols). In a nonsymbolic language, the signs are not instances of equivalence classes obeying conventions and thus not detached from individual circumstances.

In our thesis, only symbolic languages play a role, since our focus is the application of digital computers as symbol-manipulating machines[9]. In symbolic languages, we distinguish formal and natural, historical languages depending on their conventions and the way the conventions emerged. Formal languages have explicitly formulated conventions. A formal language is usually created for a specific purpose. For example, programming languages are created for programming computers, and mathematical formalisms are created for precisely describing and solving problems (Prak, 1979).

A natural language has many and complex conventions, most of which are not explicitly formulated. Next to syntax, and semantics, the conventions of a natural language deal with pragmatics, i.e., the study of the language elements in relation to their users and circumstances of their use. As noted by Wittgenstein (1953), the users of a natural language implicitly conform to the same conventions when they belong to the same social group. However, a "complete agreement" may be illusory, if only on the ground of the dynamical character of language and language learning. As a result, the presence of subjective elements in the use of language can not be excluded. Such elements may lead to misunderstandings.

Despite the subjective aspects of the uses of language, communication in everyday life seems reasonably effective. Misunderstandings may be removed during communication, when mutual corrections are possible or otherwise.

In summary, we state that the conventions for communication in natural language have a subjective part and an intersubjective part. This subdivision has its consequences, as will become apparent in Subsection 2.2.3.

9 Since the thesis is in the area of Computer Science, it is focusing on the application of digital computers. Such computers, in the form we study them, have as their main function the manipulation of symbols (by manipulating signs, as instances of symbols), a function that has effectively been applied for supporting human communication by symbolic languages and for such tasks as arithmetic (Dietz, 1996). Although digital computers may also be provided with actuators and sensors (beyond the usual I/O tools such as keyboard and video display), and be given some degree of autonomous action, such versions of computers are not the subject of our thesis.

2.2.3 Expressing and interpreting

The subsection elaborates on the expressing of mental states and processes and the interpreting of signs, as introduced in Subsection 2.2.1.

In communication we recognise three classes of elements that are interrelated:
1. Symbols,
2. Mental states and processes, and
3. The subjects about which is communicated.

The relationships were illustrated by Ogden and Richards (1923), in the well-known semantic triangle (see Figure 2.1). The relationships are as cited below. We note that the formulations cited are classical. The description of the relationships can be fit into our description of communication, by replacing "thought or reference" by "mental states and processes", and "referent" by "the subjects of the communication."

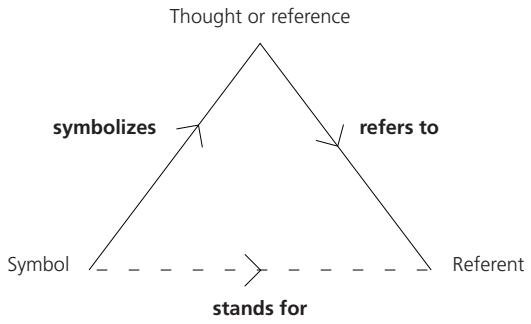

FIGURE 2.1: *The semantic triangle (Ogden and Richards, 1923).*

"Between a thought and a symbol, causal relations hold. When we speak, the symbolism we employ is caused partly by the reference we are making and partly by social and psychological factors - the purpose for which we are making the reference, the proposed effect of our symbols on other persons, and our own attitude. When we hear what is said, the symbols both cause us to perform an act of reference and to assume an attitude which will, according to circumstances, be more or less similar to the act and attitude of the speaker."

"Between the thought and the referent there is also a relation; more or less direct (as when we think about or attend to a coloured surface we see), or indirect (as when we 'think of' or 'refer to' Napoleon) (…)."

A third relationship is also recognised, on the basis of the above two relationships: "Between the symbol and the referent there is only an indirect relationship; a symbol may be used by someone to stand for a referent."

Despite later rewordings and modifications of these comments, for example by Cherry (1957), the main elements of the relationships are indicated by Ogden and Richards. Special attention is

asked for the notion of referent, which includes everything that can be referred to in communication in its so-called referential use. That is, not only "any attribute of the outside world (thing, property, event, relationship,…)" but also so-called "non-existants" and mental states and processes and their parts (Cherry, 1957, p. 110).

Ogden and Richards (1923) preserve Figure 2.1 for normal referential use of language whenever a statement is made; metaphorical use or "diplomatic" ways of stating are excluded, as well as other ways where designators are not used in their literal sense. We add that the restriction to literal use seems a simplification useful for the sake of explanation. In literal use, the relationship is established by conventions in a large group, and during a long period of time (either implicitly or explicitly made). In practice, non-literal use is also common; the convention for such use is typically established in a small group and during shorter time periods. The convention (implicitly) includes that the relationship is restricted to a specific context (e.g., physical environment, or subject area of discourse). Thus, there may be conventions for literal as well as for non-literal use of a symbol. The relationships, as depicted in the semantic triangle, are essentially the same in literal and non-literal use. In both ways of using, the conventions are partly subjective and partly intersubjective.

2.3 Information

As mentioned in Subsection 2.2.1, (human) communication serves to influence mental states and processes. The notion of information has been used in many senses, usually related to communication. Since we want to use 'information' in this thesis, in a sense useful for the field of Information Retrieval, we need to delineate what we mean by it.

In this thesis, we are predominantly interested in the support, by computers, of communication in natural language among human beings, since this kind of communication occurs in Information Retrieval. For this purpose, we provide a characterisation based on four existing definitions[10].

First, we mention the definition of information by Dietz (1993). This definition is intended for communication in natural language among human beings, possibly supported by a computer. Dietz regards information as the twofoldedness of sign and meaning; 'sign' being wider than symbol allows for communication by other than symbolic languages also. This thus allows for multimedia information as well, and is as such relevant to modern Information Retrieval.

The twofoldedness in Dietz's definition refers to the causal relationship of sign and meaning (Dietz, 1999). By 'meaning', Dietz means mental states and processes induced by the symbols according to the conventions. Dietz recognises that the conventions are meant to be intersubjective but may also have a subjective element (Dietz, 1999). However, this is not directly clear from his definition. In our characterisation, we emphasise the presence of a subjective

10 Shannon and Weaver (1949) provided a well-known definition meant for communication in a formal language with a countable number of expressions, regarding information as a measure of uncertainty. The more information is available, the smaller is the uncertainty that remains about which of the expressions of the language was received.

element in the conventions and thus in the mental states and processes induced (see Subsection 2.2.2).

Another, perhaps more important difference is that we avoid the use of the notion of meaning in our characterisation of information, since we regard it a confusing term. It has been used in several senses, and everyone seems to have a strong idea of what meaning means. For example, meaning is used by linguists for the alternative expressions of a thought, and it is used in formal logic and philosophy as truth values assigned to certain expressions (Woods, 1975; Fodor, 1998). In a third sense, described by Putnam (1973), meaning comprises of things in the head as well as elements of the world external to the head. If we use the noun "meaning", we use it in the sense of the (parts of) mental states or processes that a person expresses in a symbol, or obtains by interpreting a symbol, according to his understanding of the conventions (cf. Dietz).

The second definition of information on which we base our characterisation below is that of ISO2382-1 (1993), meant for data processing/communication. Here, information is defined as "the meaning a human assigns to data by means of the conventions applied to that data, with data being defined as a representation of facts, concepts or instructions in a formalised manner suitable for communication, interpreting or processing by human beings or by automatic means." We adopt this definition in essence, but we already mentioned our reasons for avoiding the notion of meaning. Moreover, we prefer to avoid the term 'data', since its use is frequently associated with specific applications, viz. computers as number crunchers (see, e.g., Van den Herik, 1988).

The third and fourth definitions on which we base our characterisation, are by Langefors and Samuelson (1976) and Lundeberg, Goldkuhl, and Nilsson (1981), both in the context of business informatics. They regard as information only the knowledge resulting from the symbols, and not all the mental states and processes resulting from the symbols. We regard this view too narrow, at least if knowledge is understood in our rather strict sense (which excludes mere beliefs, opinions, see the subsection below). We stretch their definitions to a wider sense that includes (next to knowledge) all mental states and processes that constitute beliefs, opinions, etc.

We emphasise that information is a mental notion, hence subjective. Since there are (usually) two different persons involved in communication (a sender and a receiver), there are also two mental states or processes, obtained by two interpretations of conventions. Thus, they are mental states or processes expressed in the symbols according to the interpretation of the conventions by the sender, and they are mental states obtained from the symbols according to the interpretation of the conventions by the receiver. In other words, there is information sent and information received. On the basis of the above considerations, we characterise information as follows.

> *Information refers to the parts of mental states and processes a human being connects with signs when expressing, or obtains from signs when interpreting them, by means of the conventions he holds applicable.*

When information is used in this sense, we thus may also say that communication serves to influence mental states and processes, and that communication serves to provide information. The latter, common use of the term information is retained here. As long as misunderstandings seem excluded, the same holds for similar colloquial expressions.

An example of information: a road sign that by convention tells traffic to stop at the railroad crossing nearby provides the passers-by that have learned the conventions for this sign with information, in the sense that it brings them in a certain mental state or process which enables them to react in an appropriate way. A person may *inform* another person about the latest stealth technology and his political ideas/opinions about the application of that technology by telling him about these issues and thereby using a natural language they both are familiar with. The result of this information process may be that the latter person believes that this technology is promising. This does not mean that he now knows that this is the case. Knowledge is different from information.

2.4 Knowledge

In analogy to information, knowledge has been defined in many ways. In this thesis, we prefer a criterion of knowledge that is compatible with our basic assumptions as outlined in Section 2.1. In the context of Information Retrieval systems, we find such a definition in Kochen (1974, p. 51). The definition of knowledge provided by Kochen is meant to be applied in the context of Information Retrieval. He states: "we suggest that an agent knows a topic if he is able to recognize and generate useful answers to questions on that topic that enable him to act purposively." (…) "Knowledge thus is a potential for a certain type of action (…)."

We adopt a slightly reformulated version of this view, in the sense that we do not require the agent to be able to generate answers on the topic. As in our example, he may act purposively just without being able to talk about it (this holds both for factual and procedural knowledge – see below). The absence of talk (or, more general: communication) may be caused by the absence of a person's capacity to speak at all. Here, it might be the case that he can perfectly speak but is just not able to express the specific knowledge. Moreover, the potential for action is wider than the capacity of speech. It also includes doing other things than speaking (or communicating in general), such as doing physical labour.

Our interpretation of knowledge matches the division by Ryle (1949) of knowledge into two kinds: knowing how and knowing that. The former comprises of skills, and the latter of so-called factual knowledge. Skills are capacities to do a certain task, involving physical acts (other than those needed for communication only). It should be noted that in this thesis we refer by knowledge to only a part of a skill, viz. the mental capacity, and not to the mere physical capacities such as muscle power[11]. (We adopt Kochen's terminology, regarding mental capacities as properties of mental states and processes).

In our characterisation, we mention performing tasks instead of solving problems. The difference between the notions problem and task is small. By using 'task', we stress that a wider interpretation is meant than the negatively connotated 'problem'. The notion of task implies a purpose; this thus needs not to be mentioned in the characterisation separately. Taking the above thoughts into consideration, we rephrase the definition by Kochen in the following characterisation.

11 We assume, as already mentioned, that mental capacity in the end is a kind of physical capacity. The term mental capacity serves to distinguish from other physical capacities.

(Human[12]*) knowledge is the mental capacity that enables a human being to perform tasks effectively*[13].

An example of a task that requires knowledge (skills as well as factual knowledge) is the solving (effectively) of a mathematical problem according to a prescribed and memorised procedure. The activity of manually building a wall of bricks also requires knowledge (predominantly skills). It also requires knowledge (mainly factual knowledge) for a person to be able of summing up by heart all the capitals of European countries.

We make two remarks on knowledge, which play a role in the thesis, in particular in Chapter 5 (on knowledge representation). They are on recognising knowledge, and on the subjectivity of knowledge.

(1) How to recognise knowledge

Since capacities cannot be recognised by observing mental states and processes, the criterion for recognition of knowledge is the effective performance of a task. It can be asked how this can be decided. In our view, this is ultimately a matter of judgement by humans, and thus at most inter-subjective. Fortunately, in many cases there will be consensus among all (competent) humans involved in the judgement. For example, it is hardly ever a problem to decide whether someone solved an elementary mathematical problem well. As another example, in sports there is usually little discussion about who reached the finish first; usually, everybody accepts the observation made by the official judge, especially when that observation is based on the finish-photograph.

(2) Subjectivity of knowledge

We already mentioned that mental notions are subjective (see Subsection 2.1.2). This also holds for knowledge. Thus, although our criterion of what is knowledge is (more or less) objective, knowledge itself is subjective. This property also plays a role in practice: different persons appear to have substantially different knowledge, and the knowledge of a person changes substantially with time (due to both learning and forgetting).

12 We limit our definition of knowledge to human beings; other mammals also have brains, and may have capacities to behave purposefully. Even machines (e.g., computers) may be ascribed purposeful behaviour. Whether such capacities are to be called knowledge, is a matter of definition, or choice, and not relevant for our thesis. We note that some AI-researchers use the notion knowledge also for computers (e.g., Newell, 1975).

13 Of course, the environment and the body of the human being also need to be appropriate, otherwise he may not be able to realise his intended acts. For instance, he cannot play the piano when locked in an empty cellar with his hands cuffed - even if he has the knowledge required to play the piano.

2.5 Documents[14]

We first characterise the notion of document, and then discuss how documents can be distinguished from each other.

2.5.1 Characterisation of document

We try to stay close to the common-sense notion of document, and to refine it where necessary. For a start, we characterise a *document* as an amount of stored signs. This corresponds to the common intuitive notion that a document is meant for communication, and that its signs are somehow captured in material substance. It also corresponds to the characterisation of document as, e.g., found in Webster's New Collegiate Dictionary (1975): "(a) a writing conveying information, (b) a material substance (as a coin or stone) having on it a representation of the thoughts of men by means of some conventional mark or symbol[15]."

According to Webster's characterisation, a document must possess permanence; hence, transient information, e.g., a sequence of sound waves, is not considered a document. Our provisional characterisation of documents is in line with this view, viz. that they *are* stored signs.

In addition, we mention a feature that stems from the proviso that a document is usually a product: a document is the outcome of a production process performed by a person or under his responsibility. The person finishes the creation of a document by declaring an amount of stored signs with some permanency to be a document. Sometimes not the creator but someone else makes this declaration; it is free to anyone to consider an amount of stored signs a document. The latter may happen, e.g., when a document is added to a library by a librarian. Hence, characteristic for stored signs to constitute a document is that it is declared a document. We thus characterise:

A document is an amount of stored signs, declared a document.

Examples of documents are: (1) a piece of paper bearing a text on it, e.g., a letter, a journal article, or a textbook, and (2) a coin that was declared a document. A counterexample of a document is a coin that was not declared a document. Also, a piece of metal having on it the contours of a human being evaporated by the uncontrolled nuclear chain reaction of 1945 in Hiroshima becomes a document when it is declared a document. The fact that it was not originally created as a document does not stand in the way of becoming one.

Although traditionally documents have had a linear structure, a document may also be a hyper document, i.e., a document having a non-linear structure created by the presence of hyperlinks. Examples of hyper documents are the documents in the World Wide Web (www); see Section 3.6. For a more elaborate discussion of documents, and hyper documents, we refer to Dietz, Wiesman, and Van der Pol (1997).

---

14 This section is based on Dietz, Van der Pol, and Wiesman (1997).

15 The Concise Oxford Dictionary (Concise, 1995) reads for document: "a piece of written or printed matter that provides a record or evidence of events, an agreement, ownership, identification, etc." The two characterisations have in common a material substance that carries signs.

In former times, documents were written, or printed matter. In this thesis we focus on a more recent kind of documents: documents to be searched and accessed by computer systems. As such, the documents are typically stored digitally[16], we call them 'digital documents'. In contrast to the single representation type constituted by a traditional document (on a fixed bearer), for each digital document we distinguish at least two types of representations: (1) representations for storing (and handling) the document (i.e., a binary representation), and (2) representations for presenting the document (i.e., representations perceptible to the senses). Of course, in most programs for editing and presenting documents, there is a one-to-one mapping between these two types of representations (barring minor details).

2.5.2 Distinguishing documents from each other

For query-based document-search processes it is necessary to distinguish documents in a collection from one another. For the latter purpose, we regard each document to consist of a *document content*, and several *document descriptors*. The content of a document refers to the document, including the representation for presentation intended by its author. A document descriptor provides additional information to the document content, such as its author and its title. The distinction between contents and additional information is common practice with printed documents.

In this thesis, we focus on content-based descriptors for distinguishing documents. In particular, we study document-search processes that use index terms for the purpose of characterising document contents.

16 Digital storage means that the stored information is encoded as binary digits (bits).

Chapter 3

Matching models and research questions

The chapter provides an overview of research in Information Retrieval, as well as research questions based thereupon. Although the overview focuses on query formulation, it treats matching and describing as well (Sections 3.1 and 3.2). These processes are discussed because they influence the effectiveness of the document-search process, and the query formulation process should be adapted to the matching and describing processes. Query formulation is discussed in three parts: the first is general (Section 3.3) and the other two focus on query reformulation with available represented knowledge (Section 3.4) and query reformulation by relevance feedback (Section 3.5), respectively. Finally, we briefly touch upon searching in webs of hyper documents (Section 3.6), since such webs (e.g., the www) have become predominant for access to digital documents. We do not discuss inspection separately but it is given some attention, especially in query reformulation by relevance feedback. The chapter concludes by providing research questions (Section 3.7), based on the overview.

3.1 Matching

The section describes the essence of matching processes. Many types of such processes exist. They all result from attempts to obtain relevance rankings of the documents in the collection with the requirement that they correspond well with the relevance ranking human beings would assign. Subsection 3.1.1 provides a general description framework for matching processes. Subsection 3.1.2 introduces the best-known matching processes by this framework.

3.1.1 General description framework

As we mentioned in Subsection 1.1.1, a matching process is defined as a process of predicting the relevance to a given query of each document in the document collection by means of symbol comparison. A matching process has as its input the descriptors of each document and a query, and as its output a relevance value of each document for the query. Since matching consists of symbol comparison, it only yields a *prediction* of document relevance for the information need of a user. In the inspection process following the matching process, the user inspects the documents

predicted relevant by matching and makes the actual judgement of their relevance[17] to his information need.

Matching processes differ in the kind of derivations they contain, as showed, e.g., in the query languages and document descriptors they use, and in the relevance rankings they produce. It is common use to call the combination of a specific kind of matching process, a specific kind of descriptors, and a specific query language a matching *model*. We provide a general description method for matching models, adapted from Lee (1994).

Every matching model can be described by:
– a collection of document descriptors (T) (e.g., indexes), used to represent documents of a collection of documents (D).
– a collection of queries (Q), being expressions that can be recognised by the matching function (see below). Q is regarded to constitute a query language. All known query languages contain keys as their basic elements. Most query languages have operators and weights as well (see Section 3.3).
– a collection of relevance values R, used to express the presumed relevance of documents in D. Often, this value is a number in the closed interval [0,1] or a part of the interval, depending on the model.
– a matching function F: $T \times Q \rightarrow R$, which assigns to each pair (t, q), with $t \in T$ and $q \in Q$, a relevance value $r \in R$.

3.1.2 Matching models

Below, we discuss the following matching models: Boolean models, vector space models, extended Boolean models, probabilistic models, connectionist models, and logical models.

Boolean models

Turtle and Croft (1992) classify Boolean models as exact-match models, as opposed to partial-match models. Exact-match models are defined as using matching functions (F) that, given a query, partition the document collection into only two sets: relevant and irrelevant documents. Partial-match models have a ranking on a larger part of the interval [0,1].

In all Boolean models, the query collection (Q) consists of Boolean queries, i.e., queries contain document descriptors as keys, next to the Boolean operators conjunction (**and**), disjunction (**or**), and negation (**not**). The document descriptors (T) are binary-weighted. The relevance of a document is derived according to Boolean logic. For example, a document is considered relevant to the query "A **and** B" only if its collection of index terms contains both keys A and B (Salton and McGill, 1983).

Since with Boolean logic alone it is difficult to cope with the linguistic flexibility of the language used in large document collections (e.g., in morphological variants), many retrieval systems use extensions of pure Boolean logic. Their query language provides proximity operators

17 Hence, at least two types of relevance are essential in Information Retrieval: relevance according to the matching process, and relevance according to the user. More types are recognised, see, e.g., Robertson (1992), Smithson (1994), and Mizzaro (1998).

and techniques for combining related word forms (for instance, a truncation operator) (Turtle and Croft, 1992). Sometimes wildcards are provided that allow for prefix, infix, or postfix parts in words. A proximity operator is a binary operator requiring that two keys always appear in a document in each other's neighbourhood. A truncation operator functions as a semi-wildcard: when positioned at the end of a string, it matches all terms beginning with the string (Salton and McGill, 1983).

Boolean models have been widely applied, e.g., in the retrieval systems STAIRS, ORBIT, and MEDLAR used in many libraries (Salton and McGill, 1983). More recently, the model is provided on many digital document collections, such as web sites, and text collections on CD-ROM.

The popularity of Boolean matching is ascribed to the high expressive power of the Boolean query language (Croft and Thompson, 1987) and the existence of efficient implementations (Salton and McGill, 1983).

Notwithstanding their popularity, Boolean models have at least three disadvantages, as mentioned by Salton, Fox, and Wu (1983).

1. The size of the output obtained to a given query is difficult to control; depending on the assignment frequency of the query keys and the actual key combinations used in a query, many documents are matched, or no document is matched at all.
2. The output obtained in response to a query is not ranked in any order of presumed relevance to the user. For example, if a query contains the Boolean conjunction ["A **and** B"], a document having only index term A (or only index term B) is considered as irrelevant as a document having none of the index terms. Also, if a query contains the Boolean disjunction ["A **or** B"], a document having both index terms A and B is considered as relevant as documents having only one of these terms. The same strict requirements hold for the negation operator and other operators.
3. No provisions are made for assigning weights to the terms attached to the documents, or to the keys of queries. Thus, all terms in the documents are assumed equally relevant to the subjects of the documents, and all keys in the query are assumed equally representative for the subjects of the user's information need.

As a fourth disadvantage, we mention that the operators in the Boolean query language are difficult to cope with; see also Section 3.3. Attempts to alleviate these three disadvantages have yielded vector space models, extended Boolean models, and probabilistic models.

Vector space models

In vector space models (see, e.g., Salton, 1991), the matching function (F) ranks the relevance on a continuous interval. Hence, vector space models are partial-match models (Turtle and Croft, 1992), as opposed to Boolean models which are exact-match models.

Regarding the query collection (Q), a query in a vector space model consists of a collection of pairs (*query key, weight*). The queries thus are without operators, but the query keys do have weights. The latter serve to indicate the importance of the keys for expressing the information need of the user. Regarding the descriptor collection (T), a descriptor in a vector space model consists of a collection of *pairs* (*index term, weight*).

The weights of the *n* descriptors of a document (*d*) are regarded as a vector: $\left(W_{d_1}, W_{d_2}, \ldots, W_{d_n}\right)$.

The weights of the *n* keys of the query (*q*) are also regarded as a vector: $\left(W_{q_1}, W_{q_2}, \ldots, W_{q_n}\right)$.

The relevance of a document to a query is calculated as the similarity of the descriptor vector of the document to the query vector. The similarity of the vectors is often defined as the normalised inner product of the vectors, according to formula (1). This formula can be interpreted geometrically as expressing the cosine of the angle between the two vectors.

$$sim(q,d) = \frac{\sum_{j=1}^{t} W_{q_j} \cdot W_{d_j}}{\sqrt{\sum_{j=1}^{t} W_{q_j}^2 \cdot \sum_{j=1}^{t} W_{d_j}^2}} = \frac{W_q \bullet W_d}{\left|W_q\right| \cdot \left|W_d\right|} \qquad (1)$$

with

q	a query vector,
d	descriptor vector of a document,
sim(*q*,*d*)	the similarity of the query (*q*) to the descriptor vector (*d*),
W_{q_j}	the weight of the j-th query key,
W_{d_j}	the weight of the descriptor in the descriptor vector (*d*) that corresponds to the *j*-th query key,
t	the number of keys in the query vector and in the descriptor vector.

The similarity of the descriptor vector to the document vector is sometimes calculated according to a different function than the above normalised inner product. Several functions were developed, e.g., the Euclidean distance (Paijmans, 1999).

Vector space models alleviate the three disadvantages of Boolean models. The number of documents retrieved can be controlled by utilising only a fixed number of top-ranked documents in the relevance ranking. Moreover, the vector space models allow for ranking the retrieved documents according to their relevance. Further, the weights to the descriptor terms and to the query keys are essential to the models. Still, vector space models are not perfect. The vector space model in its basic formulation assumes that the terms in each vector are independent. In practice, this assumption is seldom fulfilled, since the terms are related; in some cases these relationships are so strong, that they cannot be ignored (Salton, 1991).

The assumption of independence of descriptors is avoided in the Generalised Vector Space Model (GVSM) (Wong *et al.*, 1985; Carbonell *et al.*, 1997). This variant of the vector space model replaces descriptors as the basic document features with a set of 'generalised' descriptors. The descriptors are obtained from the documents; whereas in vector space models documents are usually assigned a vector of weighted descriptors (typically index terms), in the GVSM each descriptor is assigned a vector of documents and weights (representing the distribution of the descriptor over the documents). The latter vectors can be regarded dual vectors of the former, hence the GVSM is also called "dual space model" (Sheridan and Ballerini, 1996). The matching function (F) of the GVSM is similar to that of the basic vector space model, but its inner product operates on *transformed* versions of the query vector (viz., $A^t q$) and the document vector

(viz., $A^t d$). In formula 2, A_{mxn} is a matrix containing document descriptors, where *m* is the number of descriptors in the descriptor vocabulary, and *n* is the number of unique documents in the collection. The matching function (F) of the GVSM is:

$$F(q, d) = \cos (A^t q, A^t d) \quad (2)$$

The GVSM is also used for translingual retrieval. Assuming the presence of two (essentially) parallel document collections, in which one collection is the translation of the other, two matrices are formed, A and B. The matrix A is a descriptor matrix for the training documents in the language of the queries, and B is a descriptor matrix for the training in the target language, and the corresponding columns of A and B are the matching pairs of documents in the bilingual document collection. The matrix A is used for query transformation and B for target-language document transformation. The matching function is defined as:

$$F(q, d) = \cos (A^t q, B^t d) \quad (3)$$

An extension of the GVSM is Latent Semantic Indexing (LSI); here the vectors of documents are replaced by linear combinations of vectors, in order to obtain a better representative basis for the content of documents (Carbonell *et al.*, 1997).

For monolingual retrieval, a performance improvement by the GVSM over VSMs was observed on small collections (Wong *et al.*, 1985). An improved performance by the LSI over the VSM was found sometimes, but not always (Deerwester *et al.*, 1990). GVSM performed better than LSI, in contrast to what was expected (Carbonell *et al.*, 1997).

The use of the vector space model is widespread, although not as wide as the use of the Boolean model. It is applied, e.g., in the SMART system (Salton, 1991) and in the multimedia search system by Ze Wang *et al.* (1997). For a recent elaborate study of VSMs, we refer to Paijmans (1999).

Extended Boolean models

Extended Boolean models result from an attempt to increase the discrimination among the retrieved output, while still maintaining the advantages of Boolean models (Salton, Fox, and Wu, 1983). For this purpose, the document descriptors and the query keys can be weighted, and the matching function provides more values than binary. In some variants of the model, it covers the whole interval [0,1]. We discuss two variants: (a) the fuzzy-set model, and (b) the P-norm model.

(a) The fuzzy-set model

In the fuzzy-set model (Bookstein, 1980), the weight of a term is interpreted as the partial membership (of a collection or set) of that term. This idea is borrowed from fuzzy-set theory, developed by Zadeh (1965).

In the simplest variant of the fuzzy-set model of retrieval, the document descriptors are weighted and the queries are standard Boolean. Given queries "A **or** B", "A **and** B", and "**not** A", a document X with index weights $w_A(X)$ and $w_B(X)$ for terms A and B, respectively, receives the following relevance values in a fuzzy-set retrieval system:

$\max[w_A(X), w_B(X)]$ for query (A **or** B)
$\min[w_A(X), w_B(X)]$ for query (A **and** B)
$1 - w_A(X)$ for query (**not** A)

For binary document vectors with term weights 0 and 1, the fuzzy-set model yields identical relevance values as the Boolean model. Since the rank of a retrieved document depends only on the lowest or highest weighted document term for **and** and **or** queries, the discrimination among the retrieved output is still small (Salton, 1991); it is less than in the vector model, although more than in the Boolean model.

In somewhat extended versions of the fuzzy-set model, query keys may also be weighted (Waller and Craft, 1979; Bookstein, 1980; Bookstein, 1981; Buell and Craft, 1981). According to Salton, Fox, and Wu (1983) the determination of the weights is cumbersome in the fuzzy-set model. They offer the P-norm model as an alternative.

(b) The P-norm model

In the P-norm model, the relevance of a document to a query is calculated with the aid of normalised distances to fixed points. In the case of the disjunction operator, the distance to the point (0, 0, …, 0) is used; the dimension of the vector equals the number of keys connected by disjunction operators. The point (0, 0, …,0) represents the case in which none of the query keys is present in a certain document. If (only) disjunction operators are used, a document is more relevant if it is more distant from this point. In the case of conjunction operators, the point (1, 1, …, 1) is used. This point represents the case in which all documents are present in a document. If (only) conjunction operators are used, a document is more relevant if it is closer to this point.

In the P-norm model, the distances are normalised distances, according to the L_p norm. The distance for an n-dimensional vector $(d_1, d_2, \ldots, d_n)$ is defined as:

$$\left\|d\right\|_p(\textit{normalised}) = \left(\frac{1}{n}\right)^{1/p}\left\|d\right\|_p = \left[\frac{\left(W_{d_1}^p + W_{d_2}^p + \ldots + W_{d_n}^p\right)}{n}\right]^{1/p} \tag{4}$$

The symbols denote:

n the number of terms in the vector,
d the descriptor vector of a document,
the weight of the *i*-th descriptor of the document, with ,
p p-value of the p-norm.

The similarity calculations for the disjunction and conjunction operators are expressed in formulae (5) and (6). We do not provide the definition of the negation operator, nor the definition of the combination of operators.

$$sim\left(d, q_{OR(p)}\right) = \left[\frac{\alpha_1^p w_{d_1}^p + \alpha_2^p w_{d_2}^p + \ldots \alpha_n^p w_{d_n}^p}{\alpha_1^p + \alpha_2^p + \ldots + \alpha_n^p}\right]^{1/p} \tag{5}$$

$$sim(d, q_{AND(p)}) = 1 - \left[\frac{\alpha_1^p(1-w_{d_1})^p + \alpha_2^p(1-w_{d_2})^p + \ldots \alpha_n^p(1-w_{d_n})^p}{\alpha_1^p + \alpha_2^p + \ldots + \alpha_n^p} \right]^{1/p} \quad (6)$$

d	the descriptor vector of a document,
w_{d_i}	the weight of the i-th descriptor of the document, with $0 \leq w_{d_i} \leq 1$,
α_i	the weight of query key i, with $0 \leq \alpha_i \leq 1$,
$q_{OR(p)}$	a query consisting of n keys, connected by disjunction operators,
$q_{AND(p)}$	a query consisting of n keys, connected by conjunction operators,
p	p-value of the p-norm, with $1 \leq p \leq \infty$.

For p = 1, obviously $sim(d, q_{AND(1)}) = sim(d, q_{OR(1)})$. Hence, for p = 1 the distinction between the conjunction and disjunction operators disappears, and the effect of a standard vector model is obtained.

For $p = \infty$, it can be shown that $sim(d, q_{AND(\infty)}) = \min(w_{d1}, w_{d2}, \ldots, w_{dn})$, assuming that the query terms are all equally weighted. It can also be shown that $sim(d, q_{OR(\infty)}) = \max(w_{d1}, w_{d2}, \ldots, w_{dn})$. Hence, when $p = \infty$, the query-document matching function is dependent only on the document term of the highest weight for q_{OR} and the document term of lowest weight for q_{AND}. This is identical to the fuzzy set model of retrieval, and to the conventional Boolean retrieval system when both query and document terms are not weighted. For $1 < p < \infty$, a model between a standard vector model (p = 1) and a conventional Boolean model is obtained. The larger the value of p, the more importance is given to the query structure as expressed by the conjunction and disjunction operators.

Experiments by Salton *et al.* (1983) show better performances with the P-norm model than with the vector model. In three of four test collections, the improvement is some 15%, in the other collection the performances differ less than a few percents. Compared to the Boolean model and a probabilistic model (see below), the experiments show improvements of 100 up to 150%. Hence, the P-norm model seems superior to the above-mentioned models in most respects.

Lee (1994) compares several extended Boolean models: the fuzzy-set model, Waller-Kraft model, Paice model, P-Norm model, and Infinite-One model. Of these models, the P-Norm model has, theoretically, the best effectiveness.

Probabilistic models

In probabilistic models, the index terms are weighted. Each index weight of a document is interpreted as an estimation of the probability that the document is relevant to a query containing only that term (Van Rijsbergen, 1979). Hence, the weight is the probability that the index term characterises the document content. The index weights are obtained using an example set of queries, documents, and relevance judgements. In Section 3.2, methods of indexing, we mention a different manner of obtaining index weights, viz. "tf.idf"; the weights obtained in this manner may also be regarded probabilities (see Subsection 3.2.3). Therefore, probabilistic models derive their name not only from the probability interpretation, but also from the manner of establishing the probabilities.

Below, we distinguish how the weights are obtained and how they are used to calculate relevance. The weights are estimated on the basis of an example set of queries and corresponding relevance judgements for the documents in the collection. The documents are ranked according to their probable relevance to a query (Robertson, 1977). The probable relevance of a document to an individual query term may then be estimated; this yields the weight of the index term. Different approaches exist for calculating the weights. In each of these approaches, for each query the weight is based on the number of relevant documents in the collection and the number of irrelevant documents (Salton and McGill, 1983). Additionally, the term frequency may be taken into account; Kwok (1990) and Robertson *et al.* (1996) follow this approach. If no relevance judgements are available, the probabilities may be (initially) estimated by the "tf.idf" method (Turtle and Croft, 1992).

Once the weight is known, the total probable relevance of a document to a query is predicted by a combination of the relevance estimations for each key in the query. The combination is determined in a relevance function. For the relevance function there also exist different approaches. Salton and McGill (1983) use a function resembling that of the vector model, viz. as the inner product of the query vector and the index vector; the query vector is optionally weighted.

Turtle and Croft (1992) define a probabilistic model incorporating a Bayesian inference network. The query language is an extension of a Boolean language, including a sum-operator, and a weighted-sum operator; these yield added relevance values for the individual query terms. The Boolean model and the vector model can be regarded special instances of this model (Turtle and Croft, 1992). The inference network represents domain information (obtained from the document collection); the use of domain information is discussed in Section 3.5. The inference network is a directed, acyclic graph in which nodes represent propositional variables or constants, and edges represent (probabilistic) dependence relations between propositions. Given a set of probabilities for the end nodes of the directed acyclic graph, these networks can be used to compute the probability or degree of belief associated with all remaining nodes. From start node to end nodes, in successive layers, the nodes represent: an expression of the information need, queries, representations of query concepts, representations of document concepts (words), text representations from documents (e.g., indexes), and documents. In experiments on the CACM and CISI collections, the performance was significantly better than other probabilistic models, and also than the P-norm model (being a good, if not the best existing, extended Boolean model; see above).

Regardless of what weighing function is applied, probabilistic models have the disadvantage that an example set with queries and relevance judgements is needed for taking advantage of its best performance. The required presence of an example set seems a limitation for application on a large scale. Probabilistic models have been applied in research settings; see, e.g., Savoy and Desbois (1991), Callan, Croft, and Harding (1992), and Kekäläinen and Järvelin (1998). The latter authors use the model in combination with a thesaurus and automatic query expansion (see Subsection 3.4.2), and obtain a high effectiveness. This implies that the probabilistic model they used also is highly effective.

Connectionist models

In connectionist models, query terms are related to indexes of documents via an artificial neural network (ANN). Thus, an ANN determines the matching function (F). A query consists of at least

one term, and yields a subset of documents via the ANN (for an adequate introduction on artificial neural networks, see, e.g., Hertz, Krogh, and Palmer, 1991). An overview of ANNs used for matching in information retrieval is provided by Doszkocs, Reggia, and Lin (1990).

ANNs can be divided into two main types according to the way of learning: (1) ANNs with unsupervised learning (or, self-organising maps), and (2) ANNs with supervised learning. Those with unsupervised learning are applied for, e.g., the clustering of documents into groups by subject area (Lin, Soergel, and Madchioni, 1991). Those with supervised learning need to be trained before they can be used. The training occurs by providing a collection of queries, document-identifiers (or indexes) and relevance judgements to the network's input and output nodes. The ANN then, set in the learning mode, adjusts the weights of its internal connections. The weights of the index terms are not expressed explicitly; they are captured implicitly in the weights of the internal connections of the network. Below, we only discuss ANNs with supervised learning.

If the neural network has only an input layer, an output layer, and no hidden layers, the connectionist model is very similar to the vector model (Creput and Caron, 1997). However, in the vector model the terms need to be independent, whereas the connectionist model has the advantage that terms are allowed to be related. In practice, they often are related.

A disadvantage of such ANNs is that the network needs training before it can be used, i.e., it needs a collection of example queries and relevance judgements. First, it is laborious to create the collection of queries, document-identifiers and relevance judgements. This disadvantage is shared with probabilistic models. Second, the network needs a significant amount of computation time for the training (i.e., to adjust its weights).

The number of output neurons in the connectionist model equals the number of documents in the document collection. Thus, for large document collections the connectionist model requires a large number of output neurons. The number of input neurons should be in the order of the number of (distinguishing) terms in the documents, i.e., a large number in realistic document collections. As a result, a large neural network is needed, that is likely to require a large number of (document) instances to train the network. This procedure will require a large amount of computation time.

Examples of matching by ANNs with only two layers are given by Boughanem (1992) and Mothe (1994). Recently, Creput and Caron (1997) experimented with a three-layer network. In order to reduce the training time, it has incremental learning. As such, new documents can be added to the document collection without the ANN having to be retrained from the start.

Crestani (1994) reports a small-scale experiment within a collection of 200 documents. The experimental results were not encouraging: depending on the type of training that was applied on the network, they were somewhat better or somewhat less good than those of the vector space model were. So far, no literature is known on experiments with realistically sized document collections. This probably has to do with the above-mentioned disadvantage of training. More connectionist models were described in, e.g., Crouch, Crouch, and Andreas (1989), Rose and Belew (1991), and Chen *et al.* (1993).

Logical models

In logical models of matching, the matching function (F) is defined in terms of formal logic (Lalmas, 1998). Of course, the models mentioned above may also be described in such terms. In

their original specification they are not regarded logical models, since they were not defined in such terms initially.

Logical models serve as tools to specify matching processes and to predict and explain the (formal) properties of such processes. Lalmas (1998) and Sebastiani (1998) mention that logical models can be more general than the existing matching processes. This may explain why there is an increased interest in such models. An overview of logical models is provided by Lalmas (1998). A theoretical exploration of non-monotonous logic in Information Retrieval is given by Hurt (1998). Sebastiani (1998) critically reviews several issues in logic for Information Retrieval.

Logical models also allow for the modelling of the use of relations among terms by association: if a document is relevant to a specific subject, then it may well be relevant for a closely related subject. This principle is also applied in spreading activation (see Section 3.4), but that technique has developed in a less formal manner. The modelling of relations requires the presence of a knowledge representation. We discuss the application of knowledge representations in Section 3.5. Of course, the consultation of other kinds of knowledge in matching can be modelled in logical models as well (e.g., knowledge about the user).

So far, the work in logical models is predominantly explorative and theoretical (see, e.g., Smeaton, 1996; Lalmas, 1998; Sebastiani, 1998). For examples of logical models, see Crestani and Van Rijsbergen (1995), and Crestani and Lalmas (1996).

3.2 Methods of indexing

The descriptors used in the matching process are created in a process of description. Such a process is defined as a process of assigning document descriptors to each of the documents in a document collection. The descriptors are drawn from the collection of descriptors (T). Descriptors can be based on the document contents or other information on the documents, such as author, title, and date (Rowley, 1994). Descriptors of the document contents are called indexes or categories, depending on whether they are members of an indexing vocabulary or from a (subject) classification (i.e., a structured indexing vocabulary). In the remainder of this section, we focus on indexes.

Indexes can be obtained in several manners; an important distinction is that between automatic and manual processes of description. Two other characteristics of processes of description are: the kind of medium indexed, and the expressiveness of the index type. Below, we first make a remark on the medium kind. Next, we distinguish descriptor types by their expressiveness.

For many years, the known techniques of description were used in the context of text documents. With the arrival of multimedia information, there has been felt a need for effective and efficient ways of searching documents containing other media than text. One approach to such searching consists of applying a query-based document-search process. A difficult part of this approach lies in the describing process: the assignment of appropriate index terms requires new techniques for interpreting the contents of documents. As with natural language processing, these techniques are difficult to develop and have only proven successful within limited subject areas. Examples of applications are described in Feiten and Günzel (1994), Flickner *et al.* (1995), Srihari (1995), and Gupta and Jain (1997). The medium kind determines what kinds of indexing

can be used, and what performance is possible. For instance, full-text indexing does not apply to video documents; some kind of image recognition is needed instead.

Below, we distinguish index types by a characterisation of their expressiveness: simple (Subsection 3.2.1), structured (Subsection 3.2.2), and weighted indexes (Subsection 3.2.3).

3.2.1 Simple indexes

Depending on the way the simple indexes are obtained, we subdivide them into full-text indexes and conceptual indexes.

a. Full-text indexes

In full-text indexing, the set of index terms consists of the appropriate words taken from the full content of the document. Full-text indexing by definition performs the first of the following three steps (Van Rijsbergen, 1979; Salton and McGill, 1983), and usually also involves the other two steps:

– identifying the words occurring in the text of the document,
– eliminating from these words the words occurring also in a *stop list* (stopwords),
– isolating the word stems from the remaining words (by removing prefixes and suffixes).

The identification of the words occurring in the text of a document is straightforward. For the elimination process, a stop list must be available. This is a list of words not fit for distinguishing documents. According to Zipf's (1949) law, the distinguishing power of a word in a document is approximately inversely proportional to the frequency of that word in the document collection. Hence, words occurring in nearly all the documents are not suited to distinguish documents. To enlarge the recall word stems are used as indexes, instead of the full words. For isolating the word stems, lists with prefixes and suffixes are used (Salton, 1991).

Full-text indexing is easily automated, and thus allows for cheap and fast indexing of large document collections. Unfortunately, the effectiveness of document-search processes with full-text-indexed documents is poor (Blair and Maron, 1985). This is ascribed to the fact that many of the obtained index terms do not properly indicate subjects of the document. Variants to full-text indexing are developed, such as *n*-gram indexing (Chudáçek, 1984) KWIC/KWOC-indexing[18] (Salton and McGill, 1983). *N*-grams serve as descriptor strings with a length of *n* characters: all substrings of the entire document text with a length of *n* characters serve as descriptors. For well-chosen values of *n*, *n*-grams have a high distinguishing power, hence allow for effective matching. A disadvantage is sometimes that it is difficult for a user to formulate an appropriate query, since *n*-grams do not coincide with the notions of a user's information need. In KWIC/KWOC indexing a descriptor is a word accompanied by other words of the same paragraph of the document; these provide contextual information.

Document structure and document length affect the quality of full-text indexes. By selecting index terms from well-chosen parts of a document, their quality can be improved (Van Rijsbergen, 1979, and Salton and Buckley, 1990a).

---

18 KWIC abbreviates Key Word In Context; KWOC abbreviates Key Word Out of Context.

b. Conceptual indexes

Conceptual indexing is the technique of assigning index terms to a document that designate concepts. Usually, the terms are taken from a controlled vocabulary.

With conceptual indexing it is easier to obtain high search effectiveness than with full-text indexing. In full-text indexing, usually a large number of index terms is assigned to the document, many of which do not indicate the real topics of the document. With conceptual indexing, all index terms are supposed to indicate essential topics of the documents, barring errors. Since for conceptual indexing a controlled vocabulary is used, there is less variety in and ambiguity of designations than with full-text indexing. Moreover, if the controlled vocabulary is detailed, it allows for precisely indicating the topics of the documents (Rowley, 1994).

Despite the advantage of high effectiveness, conceptual indexing has the disadvantage of being difficult to formalise. For this reason conceptual indexing is usually done manually, which makes it a time-consuming and laborious process. Formalising the conceptual indexing process requires, among other things, the capability to recognise mechanically the proper meanings of words in the document. So far, this task cannot be performed adequately. It is extensively studied in natural language processing. A task successfully performed already is automatically extracting noun phrases from natural-language texts, for the purpose of indexing, see for example Evans *et al.* (1991) and Jing and Croft (1994).

3.2.2 Structured indexes

Full-text and conceptual indexes mention individual concepts. Several attempts are known for obtaining indexes with a higher expressive power, by including more than a single concept in the index, and sometimes also by relating the concepts. Here, we call such indexes structured. In general, the creation of structured indexes is even more difficult to formalise than conceptual indexes.

Frame indexing (Shute *et al.*, 1986) is a technique, which assigns pairs *(role, term)* to the documents in a collection. The variable *term* denotes a concept, and *role* denotes the role of the concept. A role can indicate, e.g., whether a bank is meant as credit provider or as savings holder. Other examples of frame indexing can be found in Arents and Bogaerts (1993) and Rama and Srinivasan (1993).

Recently, Gaizauskas and Wilks (1998) gave a wide overview of the area of information extraction. In this area, attempts are made to extract information from texts in order to fill structured representations called templates (e.g., scripts in the sense of Schank and Abelson, 1977). Examples can be found in Smith *et al.* (1989), Hahn and Reimer (1988) and Pietrosanti and Graziadio (1997). Only few experiments are known on structured indexing. This is perhaps the result of structured indexes being difficult to create.

3.2.3 Weighted indexes

Both simple and structured indexes can be weighted. The weighted simple indexes consist of pairs *(index term, weight)*. The variable *index term* indicates an index term, and *weight* to what degree an index term represents a subject of the document. The weight has a value out of the interval [0,1]. When matching in the vector model takes place, the weights are treated as a vector (see Subsection 2.1.2). In structured indexes, any element of the index can be weighted.

The predominant approach to determining the weights of index terms in text documents is

the "tf.idf" technique, a statistical technique based on the full document content which can be executed automatically. In this approach, each weight is calculated as the product of two components: the frequency of occurrence of a term in a document (term frequency – abbreviated to "tf"), and the inverse of the number of documents in the document collection to which the term is assigned as index term (inverse document frequency – abbreviated to "idf"). The term frequency is assumed to be a (preferably normalised) measure for the occurrence frequency of the term in the document. If the term frequency is not normalised, longer documents have a better chance of being matched than shorter documents, only because they contain more terms, and not because they deal more with the subject than other documents. The inverse document frequency indicates how well a term is able to distinguish a particular document from the remainder of the collection. The "tf.idf" function assigns high values to terms that occur frequently in individual local environments, and at the same time occur relatively rare as a whole (Salton, 1991).

In summary, various types of indexes have been developed, unstructured and structured ones, unweighted and weighted ones. Full-text indexes can be easily created automatically, either or not weighted by the "tf.idf" technique.

3.3 Query formulation

A query-formulation process results in a query, i.e., an expression in a formal language that can be handled by a matching process. The expression is meant to express the user's information need. As already mentioned in Subsection 1.1.1, each query is matched against the document descriptors of a document collection. Each query is an element of the query collection Q mentioned in the Subsection 3.1.1.

Query languages already played a role in our discussion of matching models (Section 3.1). Here, we summarise the main characteristics of query languages, and provide a first introduction to the processes of formulating and reformulating.

Characteristics

We discuss three characteristics found in existing query languages: keys[19] and operators (Kekäläinen, 1999). All known query languages allow for keys. A key is matched against a single, specified document descriptor by the identity relation, and thus precise matching is allowed for. The second characteristic is that operators are defined on keys. Operators increase the possibilities to specify the information needed precisely. The third charachteristic is that keys may have weights. A weight is a number, indicating the importance of the corresponding key.

Examples of unary operators are the truncation and the negation operator of Boolean models (see Subsection 3.1.2), and the weights for indicating the importance of keys. Examples of binary operators on keys are the conjunction and disjunction operator of Boolean models. In contrast to the Boolean query language, vector space models have a query language that only enumerates keys.

19 We introduced 'key' in Subsection 1.1.1. Instead of 'key', 'term' is often used. In this thesis, we reserve 'key' for elements of the query language, and 'term' for indexes. Their connection is that a key is often instantiated by a term.

The relationship between the query expressiveness and the performance has been studied by many researchers. It seems reasonable to expect that if two search processes differ only in the query languages (and of course in their matching processes as far as demanded by the differences in the query languages), the language with the most expressive query language has the best effectiveness. It is generally accepted that weights increase the effectiveness of a query language in comparison with unweighted queries (Turtle and Croft, 1991, p. 220).

In line with the above expectation, it is reasonable to expect that the Boolean query language is more effective than the enumeration of descriptors/keys allowed for by the vector space models. Hull (1997) supports this view. In experiments with a TREC collection and 25 queries, the (basic) vector space model yielded a lower effectiveness than a weighted Boolean model (both models offer – identical – weights and are identical in that sense). These results hold for monolingual retrieval and for cross-lingual retrieval (in which a query contains terms in a different language than that in which the information need was expressed). The difference in effectiveness was ascribed to the presence of the AND- and OR-operators in the Boolean query languages, and the absence of these operators in the vector space model. In monolingual retrieval the difference was approximately 10%, in cross-language retrieval the difference was approximately 35%.

Extending Boolean query languages by non-Boolean operators may increase the performance even more. Truncation and proximity operators are generally known for this property. Additionally, a sum operator, a weighted-sum operator, and a synonymous-term operator are offered in the retrieval system Inquery (Turtle and Croft, 1991) that implements a probabilistic matching model with a Bayesian network. The sum-operators yield averages of the probabilities of the terms they operate upon; this is an alternative for the AND-operator, which yields the product of the probabilities of each of the keys it connects. The synonymous-term operator yields allows for the proper treatment of equivalent terms, such as synonymous terms. Kekäläinen and Järvelin (1998) obtained an increased effectiveness by using the latter three kinds of operators in an adequate formulation procedure (see Chapter 8).

Experiments have shown that initial query formulation is still far from perfect. Qiu and Frei (1993) found that the performance of an information retrieval system is typically proportional to the length of the queries. They drew the conclusion that long queries are capable of providing sufficient information for the system to perform reasonably well. Short queries, especially vague and ill-formed ones, do not provide sufficient information and are therefore prone to poor retrieval performance.

Jansen *et al.* (1998) discovered that most users of search engines on the www create short queries. They studied Boolean queries submitted to one of the most used search engines on the www, viz. Excite. This study covered 18,113 users who submitted 51,473 queries. About 80% of the queries contained 1, 2 or 3 keys, and 62% 1 or 2 terms, 31% 1 key. The users were not aware of the fact that their queries were to be used in a research projects. Jansen *et al.* (1998) drew the conclusion that, on average, www queries are short. They mention that in studies of regular information retrieval systems, on average the queries are much longer. They give as a plausible explanation for the difference, that in such studies the users are explicitly asked to create queries as good as they can.

Another finding of Jansen *et al.* (1998) is that Boolean operators are used very sparingly. Only 6% of the 18,113 users used any of the Boolean capabilities, and these were used in less than 10% of the 51,473 queries. In those, **and** was by far the most frequently used operator. A very small percentage of the queries contained **or** or **and** **not**; these queries were made by a small fraction of the users. Moreover, only about 1% of users and 0.5% of queries created nested expressions (by using parentheses). The ability to create phrases was also seldom used – by only 6% of the users and in 6% of the queries. Thus, the possibility of creating detailed queries, as offered by Boolean expressions, is not used.

A different problem in query formulation lies in the mistakes made in terms and operators. In the above-mentioned research project, Jansen *et al.* (1998) found that a surprisingly high number of incorrect uses or mistakes was made: 50% of the users went wrong in Boolean **and**, 28% in **or**, 19% in **and** **not**. Moreover, 32% of the queries with the operator **and**, 26% with the operator **or**, and 37% with the operators **and** **not** were incorrect. The numbers may be slightly off (since 'and' is counted as error if it is not capitalised, although it may be correct in a phrase. Its correct and wrong use cannot be distinguished in these cases. From this high error-rate, Jansen *et al.* (1998) concluded that "(…) Web users are not up to Boole (…)". It seems to us that this becomes more important if a query is longer, as required to express the information need precisely.

Based on the above experimental results and our theoretical understanding, we report the following difficulties in query formulation:

1. A user has the following problems (in expressing his information need adequately in a query):
   a. Using the appropriate operators, in a proper way,
   b. Conceiving the subjects of the information need,
   c. Designating the subjects properly (i.e., in correspondence to the document descriptors).
2. The user does not know whether the subjects of the information need are present in the document collection. In other words, he does not know whether the document collection can satisfy the information need. Thus, there may be a difference of subjects in the document collection and in the information need. Such a difference has also been called a conceptual gap (Matthijssen, 1999).

The processes of formulating and reformulating a query

Query-based document-search processes usually formulate several queries iteratively for each search task. The reason for the repetition is that the initial query often yields a poor effectiveness; apparently, at the moment of formulating the initial query, the user has insufficient knowledge for formulating an effective query. We discuss this difficulty of formulation in Section 3.7 (research questions); it is the basis for our research.

The reformulation can be realised with the aid of information obtained from consulting domain information (i.e., information on the subjects of the domain of interest), and/or by information from relevance feedback (Efthimiadis, 1996). Query reformulation by consulting domain information is discussed in Section 3.4, and by relevance feedback in Section 3.5.

3.4 Query reformulation with domain information

We first discuss three existing types of domain information (Subsection 3.4.1) and then introduce techniques that have been applied for query reformulation (Subsection 3.4.2).

3.4.1 Domain information

Domain information consulted for improving a query is based on the document collection or (directly) based on human knowledge (Efthimiadis, 1996). For the purpose of information retrieval, we briefly introduce domain information based on the document collection, and two kinds of knowledge representations based on human knowledge. The latter two are thesauri and semantic networks[20].

Information based on the document collection

Reformulating a query with information based on the document collection has the advantage (over using other information) that the terminology and subjects used coincide with that in the document collection. This helps avoiding the problems 1c and 2 of Section 3.3.

With natural-language processing techniques, it is possible to extract noun phrases from text documents (Evans *et al.*, 1991). Additionally, statistical techniques can relate the extracted noun phrases (Jing and Croft, 1994)[21]. The relations can be weighted, the weights indicating a degree of relatedness of the phrases. We did not find publications on techniques for automatically identifying the types of the relations detected.

Thesauri

A thesaurus contains terms and binary relations among the terms. The terms are the designations of concepts, and the relations denote relationships between the concepts designated. The relations have names, designating the kind of relationship. The structure of thesauri is standardised, in ISO 2788 (1986). The kinds of relations allowed for are: *synonymous term*, *broader term*, *narrower term*, and *related term*. The synonymous-term relationship relates synonymous terms. The broader-term relation in general relates a more general term to a more specific term, e.g., a superclass to a subclass in a class hierarchy. The narrower term does the reverse. The related-term relation denotes that two terms are closely related. Some thesauri contain only terms, and the broader-term and narrower-term relations; such a thesaurus is also called a hierarchical concept graph, or a hierarchical thesaurus (Rada and Bicknell, 1989).

Semantic networks

A semantic network is a collection of nodes and links. The nodes stand for concepts, and the links for binary relationships between concepts (see, e.g., Quillian, 1968; Lindsay, and Norman, 1972; Maida and Shapiro, 1982). The binary relationships may be of any kind, and thus a wider variety is allowed for than in a thesaurus. Semantic networks obtain their name from the network structure they show. Although the above (general) definition of semantic networks covers the

---

20 Chapter 5 discusses domain information in more detail.

21 Statistical techniques have also been used for relating words in text documents, e.g., in probabilistic matching models and the GVSM and LSI (see Section 3.1).

generally accepted notions of semantic networks, different opinions exist what should be the precise structure of semantic networks, and the meaning of the information in semantic networks (see, for example, Brachman, 1979; Maida and Shapiro, 1982).

3.4.2 Reformulation techniques

The various reformulation techniques that we found can all be regarded query expansion techniques, i.e., the reformulation essentially comprises the expansion of an existing query by adding keys. We distinguish automatic query expansion and interactive query expansion. We do not discuss manual query expansion.

Automatic query expansion

The basis of automatic query expansion is the association hypothesis formulated by Van Rijsbergen (1979): "If an index term is good at discriminating relevant from nonrelevant documents, then any closely associated index term is also likely to be good at this." The hypothesis is worked out in a large number of variants, differing in the way the relatedness is determined[22]. Below, we give a historical sketch of several attempts.

Early query expansion methods based on statistical relations among terms had little success, see, e.g., Buckley, Salton, and Yu (1982) and Smeaton and Van Rijsbergen (1983). Voorhees (1994) tests automatic query expansion on the basis of lexical-semantic relations, as present in the dictionary WordNet (see Section 5.2). She expands queries by four expansion strategies: (1) adding all synonyms, (2) adding all synonyms and descendants in the hierarchy, (3) adding all synonyms plus all parents and descendants in the hierarchy, and (4) adding synonyms plus any set of synonyms directly related to the given set of synonyms (the concepts in the request were identified manually). The test collection is a TREC collection, i.e., a large collection (approximately 742,000 documents) covering several subject areas. The effectiveness improves only a little: the recall increases, but the precision drops. In this experiment, the short requests benefit the most from expansion. The explanation given to the latter finding is that the longer requests are already somewhat better. However, even the longer (initial) queries are still far from perfect. At least, this is a plausible explanation for the finding that by relevance feedback they can be improved much more.

McCune *et al.* (1985) describe RUBRIC, a system for rule-based query expansion. The domain information is encoded in production rules, and relates terms consisting of sets of words to terms consisting of subsets of these sets. For instance, American football player is related to American football. Further, it relates terms to their lexical variants. No relation types are specified. Experiments show an improved recall; no precision is known.

Croft and Thompson (1987) propose an approach similar to that of McCune *et al.* (1985), applying the same kind of association with extended domain information. The latter consists of designations, their constituent words, the relations of a thesaurus, a "part_of" relation, and an "instantiation" relation. Unfortunately, the performance is not measured.

---

22 For example, in Van der Pol (1996) we describe two heuristic rules for expanding a query. We will not elaborate on these early attempts, since they were never tested. They served as a prelude to our case study (Chapter 4) and the two-step query-formulation approach based thereupon (Chapter 6).

Kristensen (1993) tests query expansion with a thesaurus in a Boolean matching process. The thesaurus contains synonymous terms, and hierarchically related terms and semantically related terms (without relation type names). In the experiments, (original) queries are automatically expanded by adding all the thesaurus terms directly related to one of the query terms, for each query term. On average, this doubles the recall, and reduces the precision from 62.5% to 51.2%.

In later experiments, Kekäläinen and Järvelin (1998) and Kekäläinen (1999) focus on the influence of query structure on the effectiveness of query expansion in a thesaurus. In these experiments, they use a probabilistic matching process, with a Bayesian network (the process is that of Inquery, see Turtle and Croft (1991)). Each query is a number of facets; each facet is a collection of keys. The keys in a facet contain alternative designators of identical concepts, or designators of alternative concepts. Each facet represents a part of the information need[23]. The structure is further defined by the operators used for connecting the designators within facets and for connecting facets. The query is created by substitution each concept identifier by the designation(s) of the corresponding concept (connected by an operator.

Eight structure variants are tested, varying from weakly structured queries, i.e., queries containing only averages and weighted averages of the weights of the terms designating the concept identifiers, to strongly structured queries, i.e., queries containing the conjunction operator between facets and also a synonymy operator within facets. The latter operator is similar to the disjunction operator, but slightly different in the probabilistic environment used. The query expansion variants tested are (a) no expansion, (b) addition of synonymous terms, (c) addition of narrower concepts, (d) addition of associatively related concepts, and (e) the cumulative addition of the former variants.

In these experiments the effectiveness increases. Here, the effectiveness is calculated as the average precision over 10 recall levels or 11 document cut-off values (calculation method according to Hull, 1993). Query expansion yields a significant increase of the precision (at constant recall). In weakly structured queries, the increase is absent, in strongly structured queries it is present (this is regardless of the number of facets, that lies between 3 and 5). The highest effectiveness is obtained with the largest expansion on strongly structured queries.

Query expansion in a hierarchical thesaurus is studied in several experiments, described in Rada *et al.* (1985), McMath, Tamaru, and Rada (1989), Rada and Bicknell (1989), Rada *et al.* (1989), Kim and Kim (1990), Lee, Kim, and Lee (1993), and Lee, Kim, and Lee (1994). The query expansion is based on the addition, to each query term, of its sub- and supertypes within a given distance (with distance defined as the smallest number of hierarchical steps that form a path between the two terms). The queries are Boolean, and the matching process is probabilistic. Instead of the effectiveness, the relevance ranking is measured; the rankings are very similar to rankings made by human specialists. This suggests a good effectiveness as well.

Another group of publications shares the technique of constrained spreading activation. This kind of query expansion is applied during matching and thus does not yield an explicitly shown reformulated query, but it is functionally equivalent to expansion before matching. The terms are identified in a domain representation. The identification of related terms is called spreading activation. The name is taken from Collins and Loftus (1975), who describe the

23 This structure is identical to the structure developed in Chapter 6.

phenomenon of spreading activation in the processing of information by human beings. Since the spreading stops when certain criteria are satisfied, it is called *constrained* spreading activation (CSA). For an elaborate characterisation, and a generalised description of the technique, we refer to Crestani (1997). In his generalised description, Crestani (1997) mentions four constraints: (1) a distance constraint, using the distance of a node to the starting node, (2) a fan-out constraint, using the fan-out factor[24] of a node, (3) a path constraint, using the specific path from the node to the starting node, and (4) an activation constraint, using a threshold function.

Cohen and Kjeldsen (1987) present CSA in which the domain representation contains the subjects of research areas and the binary relations expressing relationships between the subjects (e.g., "heart_disease has_setting heart"). Each document in the document collection contains the name of a funding agency, and information on the research areas funded by the agency. Each document is indexed with the subjects of these research areas. By constrained spreading activation, a user asking for the names of funding agencies in a certain research area obtains documents about the agencies in that area, and of agencies in related areas. The spreading activation stops if one of the following three conditions is met: (1) if a certain distance to the original term(s) is exceeded, (2) if a node with many related nodes is reached (for efficiency reasons), and (3) if none of certain given heuristic rules can be satisfied. An example of such a rule is: if a subject is of interest, then its superclasses are also of interest; the rule can only be satisfied as long as there are (new) superclasses.

In their experiments, Cohen and Kjeldsen (1987) show an improvement of the recall from 44% to 67%, and a simultaneous decrease of the precision from 36% to 29%. In experiments where only the first condition is used, the recall improves further to 94%, but the precision then drops to 6%. Thus, with CSA (in Boolean matching), recall and precision seem inversely related: an increase of the recall yields a decrease of the precision.

In query expansion, the domain representations may be multi-linguistic, such that query expansion yields a query fit for searching document bases in more than one language. Sheridan and Ballerini (1996) apply a co-occurrence thesaurus for query expansion, generated from a document collection in which for each document in one language there is an almost identical document available in a different language (a comparable corpus). Sheridan, Braschler, and Schäuble (1997) use a thesaurus in which the similarity of terms in different languages is represented. The thesaurus is created by statistical correlation analysis of words in a multilingual document base, in which base each document is available in two languages (a parallel corpus).

Both the above approaches obtain reasonable performance, but considerably below that for monolingual retrieval. Ballesteros and Croft (1997) mention a performance for multilingual retrieval of 42 to 68% of the performance for monolingual retrieval, indicating the order of magnitude of the results. They ascribe a large part of the loss to problems due to the translation of individual words, instead of phrases representing concepts. Translation of the individual words in the phrases yields ambiguity. Ballesteros and Croft obtain their best results of cross-language information retrieval by an improved translation of multi-term phrases, using the words in the vicinity of a term (local context analysis) for disambiguation. This was combined with an expansion of the query both before and after translation.

---

24 The fan-out factor is a measure for the number of directly related nodes.

We note that in multilingual information retrieval (or cross-language information retrieval – CLIR) the Boolean operators are used, so far. As mentioned earlier (see Section 3.3), it is possible to obtain a higher effectiveness in monolingual information retrieval by using an extended query language, e.g., that of Inquery (Turtle and Croft, 1991). This offers an increased effectiveness in multilingual queries as well (Pirkola, 1998).

Interactive query expansion

In the above techniques for automatic query expansion, the association of terms is automatically performed. A different approach is that of *interactive* reformulation of a query (Jones *et al.*, 1995). In a small-scale experiment, the latter technique shows a high precision paired with a high recall. A simple thesaurus is used as a domain representation, containing binary relations: class hierarchies and a *related-term* relation. When searching, the user is offered all terms related (by a binary relation) to a given query term. The user then has to decide, for each term offered, whether it is to be added to the query. Although this approach seems promising, it is still an open question whether the same high recall is obtained in full-scale experiments. Further, the experiments show that, on average, the users select only 10% of the terms offered (Jones *et al.*, 1995). As the user has to read and consider each term offered, the reformulation may become a laborious task. The reformulation may become even more laborious if, in addition to the directly related terms, indirectly related terms are offered as well. In the latter case, the fraction of terms actually selected lies below 10%. This is based on the general rule that the more distant a term is, the less probable its relevance becomes (Cohen and Kjeldsen, 1987). Thus, although interactive query reformulation may be an effective approach, its efficiency is questionable.

Section conclusion

On the basis of the literature reviewed in this section, we may safely conclude that automatic query expansion has been extensively studied and tested. In general, tests results show an effect of recall enhancement. However simultaneously the precision drops, sometimes drastically. Only the most recent results show a smaller decrease (Ballesteros and Croft, 1997) or no decrease at all (Kekäläinen and Järvelin, 1998); in their experiments, partial matching was applied instead of Boolean matching.

A precision drop may result from the addition to the query of keys that, when matched in separation from the original query, yield a lower precision than the original query does. On the basis of Section 3.3, we know two reasons for such lower precision: (1) the new keys may not reflect the subjects of the information need, or (2) the new keys do not fit the vocabulary used in the documents as the original keys do. An increased precision can be explained by the opposites of the same two reasons. Thus, it seems safe to assume that the above-mentioned heuristic rule on which automatic query expansion is based will not (systematically) improve the precision if the original query does not contain precise information on the information need.

In interactive query expansion, the user is enabled to provide additional information, viz. on the relevance of expansion terms. The finding that the interactive approach yields better results than the automatic approach is in support of the above explanation: in the interactive approach the user is allowed to improve the information in the query on his information need. The finding that relevance feedback yields a much improved effectiveness, as reported in the next section, is also in support of the view that the effectiveness is high only if the subjects of the

information need are expressed in the query.

3.5 Query reformulation by relevance feedback

As mentioned in Subsection 1.1.1, relevance feedback is the utilisation of information obtained in the inspection process for revising the query matched. Relevance feedback offers information on (at least) the subject domain, the document collection, and its document descriptors. Relevance feedback thus may provide insight into (1) whether the user's information need can be satisfied by the documents in the collection, and (2) what keys are needed to express the information need adequately.

In manual query reformulation by relevance feedback, a user inspects the documents matched, learns what terms occur in the relevant documents, what terms occur in the irrelevant documents, and then adapts the query. The reformulation requires (a) a relevance judgement, and (b) revision of keys in the query on the basis of the information obtained. Spink (1994) and Spink and Saracevic (1997) study manual relevance feedback. They compare a thesaurus to relevance feedback as a source of terms for query expansion. In comparison to the thesaurus, relevance feedback yields only a few query keys. However, the latter contribute relatively much to the effectiveness gain of the revised query.

In semi-automatic reformulation, the user still determines document relevance, but the query revision is done automatically. Automatic relevance feedback was studied intensively. Below, we take the contributions by Rocchio (1968), Smeaton and Van Rijsbergen (1983), Salton, Fox, and Voorhees (1985) as a starting point for our overview.

Rocchio (1968) bases semi-automatic reformulation on the vector space model. After matching and relevance judgement, the query vector is revised as follows. A first correction vector is added to the initial query vector. The correction vector contains terms of the documents matched and considered relevant. A second correction vector is subtracted. The terms of the second correction vector are obtained from the documents matched and considered irrelevant. The results of experiments on this technique are not conclusive.

In an improved version (Salton and Buckley, 1990b), the correction vectors are weighted, by a factor out of the interval [0,1] (see formula (7)). This weighting procedure yields an improved recall. In many cases the query converges to the average index vector of the relevant documents, quite apart from the average index vector of the irrelevant documents. Whether the query converges, depends on the initial query vector. In some cases, the query converges to an average index vector with a certain distance to the relevant documents; this is called query drift.

$$q_i^{new} = \alpha q_i^{old} + \beta \frac{1}{|\text{reldocs}|} \sum_{reldocs} t_i - \gamma \frac{1}{|\text{irreldocs}|} \sum_{irreldocs} t_i \qquad (7)$$

q_i	the weight of the i-th query key in the query vector,
α, β, γ	fixed weighing factors in the interval [0;1],
t_i	the weight of term i,
$\|reldocs\|$	the number of relevant documents in the document collection,
$\|irreldocs\|$	the number of irrelevant documents in the document collection.

The above improved, semi-automatic feedback technique is tested on the Boolean model and an extended Boolean model, the fuzzy-set model (Salton, Fox, and Voorhees, 1985). In the Boolean model, they obtain an average improvement of the precision of 51%, compared to the original precision. In the extended Boolean model (the fuzzy-set model), they find an average improvement of the precision of 22%, compared to the original precision. These numbers are values averaged over fixed recalls of 25%, 50% and 75%, and averaged over four document collections[25]. In second and third revisions of the queries, the precision in the Boolean model further improves with 16% and 15%, respectively; in the extended Boolean model these values are 16% and 12%, respectively[26]. These results thus show considerable improvements.

Buckley *et al.* (1994) apply a modified Rocchio-approach in the vector space model, i.e., a modification of the above approach according to formula (6). In the latter approach, only the known irrelevant documents are included in the formula. In the modified approach, all documents left unjudged are considered irrelevant and subsequently included in the formula. In an experiment on a TREC collection, it is found that when the expansion increases, the recall-precision effectiveness (i.e., the precision averaged over several given recall levels) increases as well, viz. by 19 to 38%.

Mitra, Singhal, and Buckley (1998) further refine the modified Rocchio-approach in a vector space model (the SMART system). In an attempt to construct a technique for entirely automatic relevance feedback (i.e., with no manual relevance judgements by the user) a small set of documents matched on the original query is assumed to be relevant (without any intervention by the user). Usually, these are the top-ranked documents. Of course, the assumption is based on the general experience that the top-ranked documents are more likely to be relevant than other documents. These documents are then used to produce query terms for the expanded query. This is known to lead sometimes to query drift, especially when a large number of the matched documents assumed relevant actually is irrelevant (i.e., when the initial query does not at all express the information need, or when the matching process has a very poor performance). Instead of using the top-ranked documents for adapting the initial query, Mitra, Singhal, and Buckley (1998) first re-rank the documents matched and then apply the modified Rocchio-approach. The initial query contains all terms from a TREC request, connected by disjunction operators; this leads to a query yielding many documents, including many irrelevant ones.

Two types of reranking are compared:

1. reranking by a manually-created Boolean-like filter (including the conjunction, the disjunction, and fuzzy-set operators), that separates groups of concepts, i.e., provides a structure,
2. reranking by an automatic approach, that uses term correlation information for adapting the query; the information is obtained from networks of terms created on the basis of the document collection (by statistical techniques).

---

25 These collections are the Medlars, CISI, CACM, and Inspec; these are small standard collections.

26 The diminishing improvement by relevance feedback, for Boolean matching, was already predicted in Salton and McGill (1983).

The latter type of reranking allows for automatic relevance feedback while the former is only semi-automatic. In the experiments, the precision increases up to approximately 25%, at constant recall, in both approaches (the automatic variant being less effective as the semi-automatic). This shows that an effective automatic approach for relevance feedback is feasible.

From the above we conclude that the effectiveness of a query-based document-search process can be greatly improved by relevance feedback, without the problem of precision loss as in query reformulation before matching occurs (see Section 2.4). However, in automatic reformulation the improvement was only obtained when the initial query expressed the subjects of the information need already. In semi-automatic reformulation this was not necessary. It seems reasonable to assume that the interactive nature of the process allows the user to improve the expression of the information need.

3.6 Searching in webs of hyper documents

The section first describes the largest existing web of hyper documents, the World Wide Web. Then, it discusses research towards using the hyper structural information of webs of hyper documents.

3.6.1 The World Wide Web

The Internet hosts a distributed system consisting of a hyper-document base (a 'web' of documents), and access programs for the documents. This system is called the World Wide Web (hereafter abbreviated to www) (Berners-Lee *et al.*, 1992). In many companies and institutions, communication networks exist that are meant only for persons related to the company or institution; these networks are called intranets. Some intranets host systems identical to the www. Below, we discuss searching on the www and on similar webs of documents on intranets.

The www consists of a large number of servers and even more clients. A server is a program, which passes documents from the site ('web pages'), when so commanded by a client. A client is a program, which presents the passed documents on the client's site. The commands for passing documents are given by a user of the client program, and sent to a server program. The documents are hyper documents; a link may refer to a document on any server. The client program is transparent with respect to different servers, i.e., it offers the same shape and commands when documents are presented that are retrieved from different servers, even if the servers are running on computer sites with different types of operating systems or different communication protocols. Due to the transparency of the user interface, a user does not have to realise that he is accessing a distributed collection of documents instead of a single large document base. The distributed nature only becomes apparent through the different access times of different document bases. The differences come from the communication capacity of the communication links and from the degree of utilisation of the communication links (Berners-Lee *et al.*, 1994).

The servers and clients of the www conform to three conventions. These are (1) an address system, (2) a network protocol, and (3) a markup language. These conventions were drafted by Berners-Lee *et al.* (1994) and adopted by the developers of software for the www. The conventions are still evolving; in particular the markup language is given additional functionalities (see, e.g., Herbst, 1996).

The address system serves to assign each atom in the www a unique identification: the network address. The address system is the same as used for other Internet public domain areas, and the addresses are called Universal Resource Locators (URL). The address is a string containing (1) the address of the server on which the atom is stored, and (2) a substring to be passed to the server by a client program. The substring syntax is according to the conventions of the particular server.

The network protocol prescribes in what manner a server and a client should interact when exchanging information (e.g., passing an address substring from client to server). The protocol supports a wide variety of atom (document) formats. It is called the HyperText Transfer Protocol (HTTP).

The markup language is a language for the creation of hyper documents. It is called HyperText Markup Language (HTML). It allows for including multimedia information in the documents and for structuring the documents with, among other things, headings, lists, and menus. The documents at present in the www contain two kinds of information: (1) the information found in traditional documents and hyper documents, e.g., structured in chapters, sections and subsections, and (2) information for the support of searching by browsing, consisting of option lists.

The hypertext nature of the www was introduced for offering information via browsing (Berners-Lee *et al.*, 1994). Since browsing always starts from a document, the client program must be provided with at least the URL of that document. In practice, often some documents relevant to an information need are found after following only a few links. However, the www is not so structured that all (the URLs of) relevant documents can be found by starting from a single document. In many cases, the user finds these documents only after much trial and error, or even does not find the relevant documents at all (Notess, 1995). Therefore, query-based search is also applied, in addition to browsing for finding the (URLs of) relevant documents.

In implementing query-based document-search processes on a web, an explicit index list is needed; the alternative of using the full document contents as online source of indexes is impractical. Then the web's distributed nature would yield excessively long response times to queries. For a small part of the web such an approach may be feasible (for an example, see De Bra and Post, 1995). If the information need can be predicted roughly, it is possible to pre-fetch the documents needed, and thus obtain a faster response (Chen and Wang, 1997).

For the www, several query-based document-search systems[27] are accessible, via the www. Typically, such a system can be accessed via a document (page) of a specific server. This document contains a form, in which the user may enter a query. Once entered, a query is passed to a matching program that matches and passes back the URLs of the documents matched. The URLs then are presented to the user, via the same document used for entering the query. Subsequently, the user may retrieve the documents from their respective servers (by following hyperlinks). An indexing program that automatically finds documents on the web, by browsing

27 In Internet jargon, such systems are called search engines (Dong and Su, 1997). This name is misleading, since a search engine does not perform any conversion of energy, while engines usually do.

compiles the index list in such a system. Indexing programs[28] on the www typically take a few months for indexing the entire www. Examples of query-based document-search systems offered on the www are Alta Vista, Excite, Lycos, and Yahoo (Dong and Su, 1997). Search engines on the www operate according to the same techniques as the techniques described in Sections 3.1 to 3.5, typically offering a Boolean query language and full-text indexing. Not much has been published about the types of matching processes used. In an extensive review of a number of search engines, Dong and Su (1997) show that search engines on the www suffer from the same performance problems as other document-search systems.

In www search processes, some additional information is available for the matching process (Dong and Su, 1997). For example, domain name, location and frequency of key words in a document, and link popularity (how many other pages point to a page is an indication for its relevance; see Subsection 3.6.2). On the www also meta-search engines are used, i.e., engines having knowledge on the various search engines available. They consult for each query several other search engines and subsequently integrate the results. See, for example Howe and Dreilinger (1997), Smeaton and Crimmins (1997), and Hoogeveen (1999).

It is obvious that besides the www also other webs of hyper documents may benefit from query-based document-search processes.

3.6.2 Using hyper structural information of the document collection

The availability of hyperlinks prompts the question how query-based document-search processes may be improved by using the hyper structural information of a hyper document. We mention three attempts to do so.

(1) Savoy and Desbois (1991) and Savoy (1994) describe a hypertext Information Retrieval system with Bayesian networks in which the weights of index terms are first calculated on the basis of the "tf.idf" technique, and are then revised using link information by using co-occurrence probabilities in hyperlinked documents for the index terms. They obtain improved effectiveness.

(2) Frei and Stieger (1995) offer query-based document search in which the information in the links between the document atoms is used. The query-based document search by Frei and Stieger uses the vector space model and is extended with constraint spreading activation via document links. The performance of the spreading activation technique is tested on a Berkeley-UNIX online manual, consisting of 962 pages. This originally linear document is changed into a hyper document by the definition of hyperlinks after references that were already present in the pages (references of the kind "see also"). The hyperlinks are provided with additional information, viz. a link type, some attributes, and a free text annotation. The attributes of a link contain indexes, drawn from the documents in the source node and the destination node of the link, by a full-text indexing method. The effectiveness shows a small increase. However, when the user is allowed to adapt link attributes while iteratively searching,

28 Several names exist for these programs. Koster (1994) mentions for example Web Walkers, Spiders and Robots. We stress the fact that these programs inspect the distributed hyper document base, but do *not* move themselves over the Internet (at the time of writing), although such action may be suggested by their names.

the effectiveness increases considerably (in particular after the first iteration). Before the first iteration, the link attributes contain only information drawn from the documents in the link source atom and destination atom; after the first iteration the links also contain information added by the user.

(3) Recently, Chakrabarti *et al.* (1998) developed a technique (named HITS, after Hyperlink-Induced Topic Search) for searching for authorative web documents on the www, given a query with a sufficiently broad subject. The technique starts by normal matching, and then applies relevance feedback on the top-ranked documents, using the hyperlink structure of the documents matched. In the relevance feedback, the hyperlinks in the documents are used for extending (once) the collection of documents matched. Then, the documents in the groups are assigned (in an iterative process) two values: a *hub* value and an *authority* value. The former indicates how many documents it refers to, the latter indicates by how many documents it is referred to. The assignment is an iterative process. The resulting values provide a grouping of documents, in which each group represents an interpretation of the (general) subject of the query. Thus, not homonymous terms are distinguished, but subtypes of the general term. For instance, documents about fishing in general were distinguished from documents about fishing in the south of Medford, North Dakota. The general subject obtained the highest authority value. The grouping starts since in practice documents refer only to other documents with the same interpretation of the term (i.e., to documents having the same subject). Experiments show that the relevance ranking by this technique is closer to that by human beings than the one by the normal matching technique only. It thus may be expected that the technique yields increased effectiveness on the basis of information in the hyperlink structure.

The technique HITS is further developed in a technique named CLEVER (Chakrabarti *et al.*, 1998). Instead of using the mere presence of links, it additionally filters links by the information in their neighbourhoods on the pages that they relate (moreover, it uses some idiosyncratic properties of documents on the www). In experiments with CLEVER on the www, it was compared (in 27 queries) to Alta Vista, regarded the search engine with the largest number of pages indexed and of relatively high quality among those using automatic indexing, and compared to Yahoo, regarded the best search engine among those having a manually created subject classification. Since the www sometimes yields several thousands of documents matching a query, it was regarded too laborious to evaluate the recall. Instead, for each query a fixed number of documents matched were evaluated for relevance, and as such the relative precision values of the three search engines could be established. CLEVER outperformed both Alta Vista and Yahoo, the latter being second best; the average precision over all 27 queries was approximately 50% for CLEVER, 35% for Yahoo, and 15% for Alta Vista (Chakrabarti *et al.*, 1998).

In summary, the hyper structure of a hyper document sometimes provides useful information to the matching function (F) for improving the effectiveness of query-based document-search processes considerably.

3.7 Conclusions, a problem statement, and three research questions

The section summarises the main conclusions of the literature overview. On the basis of the conclusions, a problem statement and three research questions are formulated.

3.7.1 Conclusions

In our overview of previous research, as presented in Sections 3.1 to 3.6, we found that much attention has been given to the matching process. Several matching models have been developed, some very effective, and still reasonably fast.

The process of description has also obtained considerable attention: advanced types of (structured) descriptors have been developed. Although some descriptor types are promising, their performance has hardly been tested. Unfortunately, it is difficult to write programs that create advanced descriptors automatically. This is still a research challenge. In practice, we experience that full-text indexing is the standard process of description. Often the indexes are weighted, using the "tf.idf" technique.

Query formulation has obtained relatively little research attention, as far as the formulation of an initial query is concerned. Yet, the formulation of the initial query can hardly be called successful: the effectiveness of this query is very low. Two known causes for this problem are that the user does not adequately express his information need in the query, and that unexpectedly many errors are made when using operators.

Query reformulation has obtained much attention, virtually always in the shape of query expansion. Query reformulation on the basis of domain information has usually led to an increased recall and a decreased precision. In the best case, the precision increased somewhat. Query reformulation by relevance feedback has given considerable increases in the effectiveness of the query reformulated, in comparison with the initial query (here its effectiveness means both the recall and the precision). It seems to be a condition for successful query reformulation by relevance feedback that the initial query expresses the subjects of the user's information need fairly; if it does not, query drift occurs.

Hyperlinks have successfully been used to increase the effectiveness of matching. However, on the basis of the results so far, it is improbable that hyperlinks will boost the performance of query-based document-search processes almost to perfection. Hyperlinks remain in the first place tools for browsing.

3.7.2 Problem statement

For Boolean matching with full-text indexing, it is in principle possible to formulate a Boolean query that offers a 100% recall and precision. Blair and Maron (1985) already suggested this. Such a query can be constructed in retrospect, provided that all relevant and irrelevant documents of the document collection are known. Although this construction method is of no practical use, the lesson that a query yielding a very high precision and recall is possible provides the above useful insight.

The expectation that better queries are possible (by an improved query formulation process) is confirmed by the finding that relevance feedback, as described in Section 3.5, yields significantly improved queries. An obvious explanation for improvements obtained in relevance feedback is that the user obtains knowledge, which then is used for making a new query. One could of course rely on relevance feedback only, for obtaining an improved query. However,

relevance feedback is time-consuming. Moreover, the knowledge gained in relevance feedback has a heuristic nature and may well be only a part of the potentially useful knowledge for the process of making a query.

We regard it as our challenge to improve the effectiveness of the initial query formulation process. Our focus is the use of represented domain knowledge for formulating an initial query (i.e., a query before the first time of matching). Our approach distinguishes itself from existing approaches in that it will start at an earlier stage in the formulation process, i.e., before a query is formulated at all. This inspires us to our main research question, or problem statement:

> *How can, in query-based document-search processes, the effectiveness of the formulation of an initial query be improved by utilising represented knowledge?*

The problem statement is worked out in three research questions. We note that by focusing on the formulation of an improved initial query, we do not reject relevance feedback as a means for (further) improving a query; we believe that both approaches can be used next to each other, and may yield better results than each approach in separation. However, the research reported focuses on the improvement of the initial query only.

3.7.3 Three research questions

It is generally agreed upon that improvements are promoted by understanding. Thus, in order to improve query formulation, we may benefit from an increased understanding of these processes. Since we want to support the (initial) query formulation process with represented knowledge (procedural and/or factual), it seems useful to know how queries are formulated in practice. In patent-search tasks, invoked in the writing of a patent application document, queries are formulated more carefully than in the settings mentioned in Section 3.3 by Jansen *et al.* (1998). The formulation process may yield insights for that can be incorporated in a computer program supporting query formulation. This prompts our first research question.

> *1. How are queries formulated adequately, in a search task that serves the writing of a patent application document?*

The first research question is dealt with in Chapter 4. There, we analyse query formulation by a study in patent searching.

Moreover, our problem statement mentions represented knowledge for improving query formulation. The reason is that domain knowledge is known to contribute to the effectiveness of query formulation. As we discussed in Section 3.5, representations of domain knowledge currently used in Information Retrieval are thesauri and semantic networks. However, these known representations of knowledge, as used in Information Retrieval, seem not perfect. For instance, their meanings are unclear, and the information they contain is often undetailed (this is elaborated in Chapter 5). Another indication for their imperfectness is that, next to relevant terms, a thesaurus also yields many irrelevant terms (Spink, 1994; Spink and Saracevic, 1997). In a comparison to the thesaurus, a user in combination with relevance feedback gives a large number of useful terms. Although the cause for this result is not precisely identified, one probable explanation is that the standard thesaurus is too coarse a representation. If it contains

more refined knowledge, it may well allow for a more elaborate selection of terms. On this conjecture we base our developing of an improved knowledge representation language. Our second research question is therefore directed at this issue:

> *2. Is it possible to formulate a representation language for domain knowledge that effectively supports initial query-formulation processes?*

The question is treated in an engineering fashion. We first formulate requirements, then study existing representation languages, and finally design an adapted language (see Chapter 5).

After assessing the second research question, we are ready for the actual improvement of query-based document-search processes. We focus on matching processes with Boolean query languages, since these are widely applied languages, and they offer a reasonable expressive power. With the lessons learned from the first and second research questions, we should be able to develop knowledge-based techniques that improve the formulation of queries, and that do not have the disadvantage of the approaches discussed in Section 3.4. By 'knowledge-based' we mean based on domain knowledge as well as on task knowledge. This inspires us to our third research question.

> *3. Is it possible to formulate a knowledge-based technique that effectively supports initial query-formulation processes?*

The third research question is worked out in Chapters 6 to 8. First the techniques are developed, and then they are implemented and tested.

Chapter 4

Query formulation in patent-search tasks

The chapter provides an explorative study of query-based document-search processes as applied in a patent-search task. It thus assesses the first research question: "*How are queries formulated adequately, in a search task that serves the writing of a patent application document?*" This question is based on the expectation that regularities can be observed in query-formulation processes for such patent-search tasks. The regularities, once known, may be incorporated in software tools for supporting query formulation; it is of course the motivation for this explorative study.

The chapter is structured as follows. It contains a theoretical description of query formulation (Section 4.1), followed by an empirical study in patent searching. Patents are introduced as far as seems relevant for a proper understanding of patent-search tasks (Section 4.2). Query-based document-search processes are described, as they are applied by patent attorneys. The description serves as a setting for simulated patent-search tasks (Section 4.3). Subsequently, the simulated patent-search tasks are specified (Section 4.4). Finally, results and an evaluation are reported of query formulation in the patent-search tasks studied (Section 4.5).

4.1 Description of query formulation

In this theoretical description, we adopt the view of Belkin and Croft (1992) and Smithson (1994) that often a search task is started when a problem is encountered during the performance of some other task (i.e., during an activity with a goal). The search task then is aimed at solving the problem. The supertask invoking the search task provides the goal and relevance criterion for the search task. We characterise the setting of such search tasks as follows.

A person (or group of persons) is performing a task, and has at his disposal a collection of documents too large to be familiar with its contents, and too large to search information in an efficient way by inspecting the documents in the collection instance by instance. Information is searched by invoking a query-based document-search process. The other information and knowledge the person has at hand does not suffice to complete the task successfully. Hence, we say

that the person lacks knowledge he needs for attaining the goals in his task, i.e., he has a *knowledge lack*. This knowledge lack may be eliminated by information obtained from a sub-collection of documents in the collection. There may even exist several (possibly disjunct) sub-collections of documents, each eliminating the knowledge lack. Before the task can be completed, at least one such a sub-collection must be identified. We note that the knowledge lack exists regardless whether the person recognises it. The search task may be performed by himself or by another person.

Given the above, we define the notion of document-search task:

> *A person's document-search task in service of another task is the task of identifying a sub-collection in a document collection, which sub-collection provides a specified person with the information needed to eliminate that person's lack of knowledge which occurs during the other task.*

According to this definition of a document-search task, we define document relevance as follows:

> *A document is regarded relevant for a person in a document-search task in service of another task if and only if it is in a subcollection of the document collection consulted in the search task, which subcollection resolves the knowledge lack meant to be resolved by the document-search task.*

The knowledge lack applies to a specific person, but that person does not always precisely know the lack. He has a certain understanding of the knowledge lack, i.e., he conceives his own knowledge lack in a certain manner. Of course, the person's having a good understanding is in service of an effective document-search process, since he is more likely to search for the information that really solves the knowledge lack than for other information.

Distinguishing a knowledge lack and the understanding thereof, we define the notion of information need as follows (as introduced in Subsection 1.1.1):

> *An information need in a document-search task in service of another task is the searching person's need for information that eliminates the understood knowledge lack in the other task.*

Following Ingwersen's (1996) assessment of information needs in a document-search task in the context of another task, we assume that someone's mental states and processes have some stability, at least for a while. The knowledge lack and the information need therefore may be considered more or less constant for some time. The time periods mentioned above span at least the task and the search task (as long as no strong emotional events happen, and no information arrives essential for the knowledge lack or the information need, etc.). Without such stability, it is difficult to build a coherent theory on document-search tasks.

From the definition of a document-search task it follows that for determining the relevance of a document, the whole sub-collection needs to be identified. In practical search processes (such as those of the sections following), it is deemed too difficult to identify precisely a sub-collection of documents that eliminates the lack of knowledge. Therefore, in practice a simplified document-search task is applied:

> *A document-search task in service of another task is a task to identify individual documents, each having as subject(s) the subject(s) of the knowledge lack in the other task.*

In the simplified document-search task, the notion of relevance of a document thus is reduced from providing adequate knowledge to being about the proper subject (this is sometimes called "aboutness"). Obviously, the reason is that query-based document-search processes support searching by subject, since query languages allow for the expression of subjects. Since it is clear that most documents considered relevant in the full document-search task are about the subjects of the knowledge lack, the simplified search task still yields in essence the required collection of documents, i.e., the solution to the full document-search task.

With the above definitions, we describe the query-formulation process as part of a document-search task invoked in another task; see Figure 4.1[29]. Two steps are distinguished in the formulation process:

1. developing an understanding of the knowledge lack, and
2. expression of the information need.

The two steps may be intertwined. We draw the document-search task seperated from the other task, i.e., as a distinct task. It may as well be regarded a subtask of it, since this does not change its content.

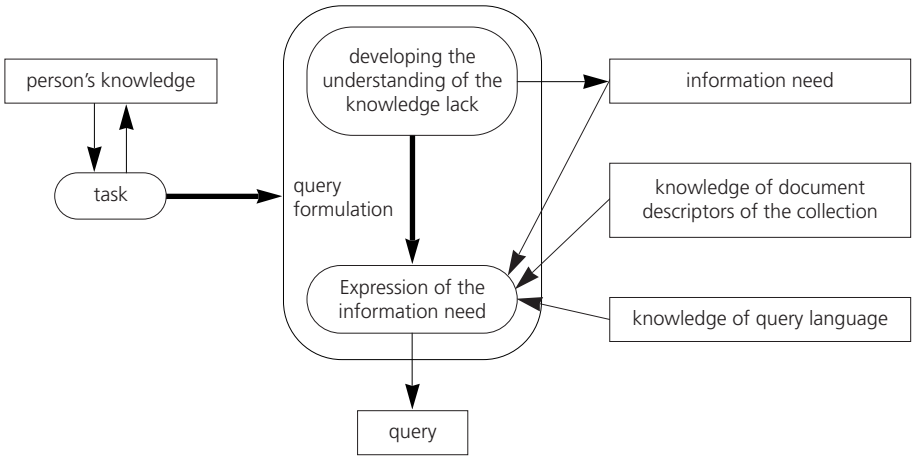

FIGURE 4.1: *Schema of the query formulation process.*

Figure 4.1 differs from the known descriptions of query formulation (see Subsection 1.1.1) in that it starts from a knowledge lack, instead of an information need. The lack itself is not shown, since the conventions of the figure do not allow for this; the person's knowledge is shown

29 Figure 4.1 is drawn according to the convention of subsection 1.1.1. Additionally, a rounded box in another rounded box indicates that the inner box represents a subtask of the outer box.

instead, and this knowledge appears to be insufficient, and thereby it constitutes the knowledge lack. In the empirical study, it will become apparent that the first step (developing the understanding of the knowledge lack) actually plays an important role in query formulation – at least in the study.

As will be clear from the above and Subsection 1.1.1, the kinds of knowledge needed in these two steps are (1) the person's knowledge, (2) the information need (or, the person's understanding of the knowledge lack), (3) the knowledge on the descriptors used in the document collection (and what they refer to), and (4) the knowledge of the query language. We note that the knowledge of the descriptors comprises knowledge of which descriptors are present in the descriptor collection of the actual document collection rather than all the descriptors allowed for, since this constitutes knowledge on what is in the actual document collection and thus reduces the conceptual gap.

The empirical study reported in the following sections provides details on the description of this section.

4.2 Patents introduced

The section provides background information in three subsections. Subsequently, we discuss (1) what are patents (Subsection 4.2.1), (2) what are patent documents (Subsection 4.2.2), and (3) the role of patent searching in the writing of a patent document (Subsection 4.2.3). Patent law is a part of intellectual property law, and is a complex legal field. Here, we only mention elements important for a proper understanding of the patent-search tasks as described in the Sections 4.3 and 4.4. The contents of this section has benefitted from Van Dijk (1994) and Stevenhagen and Van Leeuwen (1995).

4.2.1 Patents

According to Dutch patent law, a patent is an exclusive, temporary right of a legal person on an invention. An invention is defined as (1) an industrial product, that is (2) new, and (3) inventive. Industrial products can be devices, methods, and substances. An example of a device is the continuous adjustable transmission with a steel belt, applicable in, e.g., cars. An example of a method is the well-known Polaroid process for developing photographs without any liquid fluids. An example of a substance is chemical sweetener for coffee. The duration of the right constituted by a patent differs among countries. In the Netherlands, it is possible to choose between durations of six years and twenty years.

A patent is granted to a legal person, i.e., an inventor or his employer. The lawful authorities of the Netherlands do the granting. Patents are granted only on request. The request can be made by filing a patent application, in the form of a document. The request initiates a procedure (called "granting procedure"), in which the applicant has to convince the authorities that the requirements for granting are met. This usually involves several communications between the applicant and the authorities. At a certain moment in the procedure, the application becomes published (even if no patent is granted in the end). Once the patent is granted the application becomes published again, now as a patent document; the latter document may contain minor changes with respect to the initial application document. Granting procedures differ substantially

among countries; in some countries it is hardly investigated whether the requirements for granting are met; in other countries it is investigated thoroughly.

Most countries adopt the above notion of invention. Moreover, they have international treaties that make it possible to file a patent on a particular date (the priority date) in one country, and to file it in other countries at a later stage (within a year). The patent rights in the other countries then start at the priority date.

A major goal of patent law is the stimulation of technological developments. This is realised in two ways. On the one hand, patent law protects ideas, by granting a temporarily exclusive right to the inventor. The exclusive right creates a temporarily monopoly in order to stimulate the further development of the invention. Without the exclusive right, the idea could easily be copied by others, making it less profitable for the inventor to work out the idea. On the other hand, patent law makes the ideas accessible to the public. The disclosure of the ideas serves to make the new technologies accessible to others than the inventor. As such, they are enabled to apply them once the patent protection has ended, or to use them as inspiration for the development of new inventions.

4.2.2 Patent application documents

A patent application document serves to define an invention. It plays an important role in the granting procedure of a patent. In this kind of procedure, a patent application document is used for verifying whether the invention meets the above three requirements. There are two typical moments for performing such verification: one is during the writing of the patent application document (by an inventor or his attorney), and the other is during its examination, as performed by the authorities during the granting procedure.

A Dutch patent application document is divided into the following parts (Stevenhagen and Van Leeuwen, 1995): (1) description part, (2) claims, and (3) abstract. We elaborate on these parts below.

First, the description part contains (a) an introductory part, (b) a description of the prior art, (c) a problem statement, (d) a description of the solution, and (e) illustrated examples of the invention. The introductory part contains the essential but known characteristics of the invention. We regard a characteristic essential if and only if it is needed for the working principle of the invention. The description of the prior art mentions existing technology directly relevant to the invention. Usually, it cites a small number of other patent publications (typically 1 to 5). The problem statement tells disadvantages of the existing technology. The description of the solution tells the characteristics considered inventive and new. The illustrated examples are described in text and clarified by (usually technical) drawings.

Second, the claims are descriptions of the invention; they serve as primary definitions. Usually a patent has several claims, all based on the same idea, having varying degrees of specificity. The first claim describes the invention with only the most essential characteristics; hence, it is a general description of the invention. The successive claims describe additional characteristics; hence they are more specific. In some patent documents, one claim describes a method, and another claim describes a device; in such cases both claims describe a different realisation of the same idea. In the Netherlands, the claims are interpreted in the context of the description part of the

patent publication, in particular the examples. Thus, the description part and the claims together define the invention, not the claims only.

The claims have to describe the invention at an appropriate level of abstraction. The proper level of abstraction is the one at which only essential characteristics are mentioned. We regard the description at too high a level of abstraction, if it mentions only some of the essential characteristics, or mentions superclasses of essential characteristics (e.g., bearing instead of ballbearing). Then it is too general a description and may also include technology already known; as a result, the invention may not be new and a patent may not be granted. We regard the description at too low a level of abstraction, if it mentions more characteristics than only the essential characteristics, or mentions subclasses of essential characteristics. Then it is too specific a description and does not include all the variants that belong to the idea of the invention. If this occurs, a third party may create such variants; the patent does not protect the inventor against the exploitation of the variants by the third party.

An example of the first claim of an invention, illustrating that a claim is a textual (legal) definition, is the following (copied from De Jong, 1990):

> *"A method of displaying navigation data for a vehicle in an image of the local vehicle environment, characterized in that an image of the local vehicle environment is generated by an image pick-up unit, which image is displayed on a display unit, an indication signal formed from the navigation data being superposed on the image."*

Third, the abstract is an abstract in the common sense of the word. It usually is a brief version of the major claims. It contains most of the essential subjects of the patent application document, and no minor details.

We mentioned patent attorneys as writers of patent application documents. Patent attorneys are professional service-providers in patent law, qualified by the authorities. They offer general support in patent affairs, such as in the application of a patent, in the opposition against the granting of a patent to someone (typically a competitor), the asking for the nullification of an existing patent, the trade of a patent, and the granting of a license. Although inventors are allowed to handle their own patent affairs, it is common practice to hire a patent attorney for such tasks.

4.2.3 The role of patent searching

Document-search processes play an essential role in the patent application procedure. They are used in the above-mentioned two judgements whether the three requirements for an invention are satisfied. The search process serves to identify documents to be used for evaluation whether the second and third requirements are met, i.e., newness and inventiveness. Both requirements are evaluated relative to the technology known already, called the "prior art", as determined through a literature review. Newness can be determined objectively; inventiveness is determined by a subjective judgement. Guidelines have been established in order to promote equal judgement among different persons. An example of such a guideline, in Dutch patent law, is that a combination of two individually known technical measures is considered inventive only if the combination yields an unexpected effect (for skilled workers). For example, if two known types of fruit jam are mixed, the newly obtained type of jam is considered inventive only if it has a taste that is not expected directly as the mixture of the tastes of the two known types of jam

(Stevenhagen and Van Leeuwen, 1995). The search process is applied by the authorities that judge a patent application, and also by the writers of an application, i.e., inventors and patent attorneys (as mentioned above).

4.3 A patent-search task

The present section focuses on a patent-search task as performed by patent attorneys (Subsection 4.3.2). The description of the task provides a setting for the simulation of patent-search tasks as described in Section 4.4. The patent-search task is in service of, and determined by another task: the writing of a patent application. We first describe the other task (Subsection 4.3.1). This description is based on interviews with three patent attorneys[30].

4.3.1 Description of the supertask

The task of writing a patent application is performed in the practice of a patent law firm. The task starts with a request by a client (an inventor or his representative). The latter informs the attorney about the invention: he shows a physical embodiment of the invention, or a registration thereof (e.g., a picture or drawing). If there exists no embodiment yet, the inventor describes the idea (e.g., verbally, or by way of a design drawing).

Next, the attorney starts writing the patent application document. As discussed in Subsection 4.2.2, he expresses the essential characteristics, distinguishing known characteristics from new characteristics at an appropriate level of abstraction. The attorney and the inventor conceive variants to the embodiment shown or described by the inventor. This serves to develop an understanding of the invention, i.e., what characteristics are essential: the inventor usually has conceived a specific embodiment, whereas alternative embodiments, having different characteristics, would equally well suit his idea.

Usually, the attorney and the inventor do not have the knowledge needed to evaluate what characteristics are essential for the invention, and what characteristics are new and inventive. We assume that the following kinds of knowledge are needed for such an evaluation:

1. knowledge of the prior art, i.e., what technology already exists (and in which documents it is published),
2. knowledge of the working principles of (a) the prior art, and (b) the invention, and
3. knowledge of the terminology.

Obviously, an attorney cannot possibly write an application if he has no knowledge of the above three kinds at all. If he knows everything about the prior art, the working principles and the terminology he should be able to tell immediately what characteristics are essential, and which of these are new and inventive. Thus, he then has the knowledge to write the patent application. In practice, the attorney has some knowledge of each of the three above kinds. For extending this knowledge, he invokes a query-based document-search process. It depends on the budget, set aside for writing the application, how extensive the search process is performed.

30 These attorneys are affiliated with the patent law firm EP&C (located in Rijswijk, the Netherlands).

4.3.2 Description of the patent-search task

As mentioned above, the knowledge lack to be resolved by the query-based document-search process comprises knowledge of the three kinds mentioned above. Following the classification of relevance by Smithson (1994), i.e., in aboutness, usefulness, and user satisfaction, relevance in this document-search task seems related to usefulness. As discussed in Section 4.1, usefulness is substituted by aboutness as a relevance criterion, since query-based document-search processes allow for searching by aboutness. It is assumed that the useful documents are in a subset of the documents with the subject of the invention. Aboutness is determined foremost by indexes, and sometimes also on the basis of other document descriptors.

The document-search process invoked by the patent attorneys is an instance of a process type with the following main characteristics[31]:

– *The document base* (D) contains the patent documents (both applications and granted patents), essentially from all over the world and published from 1975 to the present day; only the text part is included, not the drawings.
– *The document descriptors* (T) are full-text indexes, completed with additional descriptors, such as inventor name, application date, and subject category.
– *The query language* (Q) is Boolean, offering the disjunction, conjunction, and negation operator, and additionally, proximity operators and a truncation operator; the query language allows for querying by indexes and by the additional document descriptors.
– *The matching process* (F) is Boolean, i.e., it has no relevance ranking. The matching process yields the number of documents matched, and their identifiers.
– *The inspection process* enables the user to inspect additional information on the documents matched, viz. the document content and title, the abstract, and the name of the inventor.

4.4 Simulated patent-search tasks

To obtain detailed knowledge on the search process applied in the task of writing a patent application document, we have simulated five patent-search tasks performed by a test-person[32]. These tasks are almost identical to those described in the previous subsection. In each task, the test person is provided with a brief description of a fictitious invention. Although fictitious, the inventions are inspired by real inventions, and considered realistic examples of the latter. The test person is asked to do each search task as if it were part of the task of writing a patent application. The actual writing did not take place. As a measure to obtain a realistic simulation, the tasks are done under some time pressure. The test person has 25 minutes per task, sufficient for an elaborate but not exhaustive search process. The tasks are exemplified by the specification of the topic of task 1:

---

31 This search process is offered online by the (commercial) database provider Questel.

32 The author himself, being a former patent attorney trainee of EP&C. The search tasks were done (shortly) before the author read the research literature on query formulation. As such, his scientific knowledge of query formulation was no obstacle to simulating the query formulation process that he knew it from his earlier job.

A Peltier (semi-conductor) cooling device that offers increased cooling capacity; in particular it offers the possibility to obtain a high temperature difference between the two sides of the Peltier 'panel'.

The search system has a slightly different search process type than the type described above. We chose for the search system because it was available free of charge[33]. Below, we describe the differences.

– The *document base* (D) is considerably smaller than the Questel database. It contains only the patent abstracts of the U.S. Patents published in 1994.
– The *query language* (Q) is less powerful, viz. that of WAIS (Pfeiffer, 1995). WAIS has a kind of Boolean query language. It differs from standard Boolean languages in that the negation operator may be used only at the end of a query, without using parenthesis. It thus can be applied to exclude only a single word, but not for excluding several single words. The WAIS query language also offers some natural language-like features. In the simulated search tasks these are not allowed for, in order to maintain the similarity to the original search tasks.
– The *matching process* (F) yields (1) the number of documents that matches the query, (2) the title of each document, and (3) a relevance value for each document (being a number on the interval [0,1]. The relevance ranking is not offered in the search system by Questel.
– In the *inspection process*, the titles of the document matched are presented to the user in order of decreasing relevance. In the Questel system, they are presented without a relevance ranking. Therefore, the test person is asked to ignore the relevance ranking in the simulations.

The next section presents results and an evaluation of the five simulated patent-search tasks of the present section.

4.5 Results and evaluation

We present results in Subsection 4.5.1, a discussion thereof in Subsection 4.5.2, and conclusions and suggestions for future research in Subsection 4.5.3.

4.5.1 Results

The queries as formulated in the sessions are written down, and the notes taken by the test person during the sessions are recorded in writing and audio. We provide an example of the results, by the first two queries and the last (i.e., seventh) query of the first search task. In addition, Table 4.1 shows the number of queries created in each task.

Query 1: "Peltier"

Query 2: "cooling AND device"

33 The search system and the document base were offered on the WWW by Multicasting Service in the USA.

Query 7: "(Peltier OR thermoelectric) AND (cool OR (heat ADJ transfer)) AND (device OR apparatus OR system OR assembly OR convertor OR material) AND (material OR gradient OR (temperature ADJ difference))"

Task number	1	2	3	4	5
Number of queries	7	11	2	3	6

TABLE 4.1: *The number of queries created per task.*

The queries formulated are reported on the www[34], including comments given by the test person during the patent-search sessions.

4.5.2 Discussion

We make a general remark and discuss three specific issues.

General remark

As a general remark on the results obtained in the study, we mention that in some of the five simulated search tasks the documents matched did not have topics very similar to the topics of the tasks. According to the test person, the similarity of given inventions to the topics in documents matched is higher in the real search tasks of his own experience. This finding is based on the search tasks 3 and 4. In these tasks, it seemed not possible to find documents about inventions that resemble the potential invention to a high degree. A plausible explanation for this is that there is a large difference between the sizes of the document bases used in the practice of patent searching, and in the simulated tasks. It seems reasonable to assume that the small document base, as used in the simulation, contains fewer documents that resemble the potential invention closely than the former base does.

The user's understanding of his knowledge lack

We observe that the queries formulated in the search tasks are subject to changes in both terminology and (underlying) concepts. On the basis of the theory developed in Section 4.1, we recognise four possible causes: (1) a change of the knowledge lack, (2) a change in understanding the knowledge lack (i.e., information need), (3) a change in knowledge of the document descriptors of the actual document collection, and (4) a change of the knowledge of the query language.

The first cause is not important in this study, because the documents are thrown away after each query is matched and evaluated. The knowledge of the user does not count here; he has to identify documents providing all the knowledge which is lacking at the start of the search task.

34 http://www.cs.unimaas.nl/~pol/searchtasks.htm

The fourth cause is also not important in the study, since the user in question had full command of the query language used.

The second and third causes both seem to account for the changes. Some changed terms seem part of the third cause: they are synonymous or quasi-synonymous terms. Other changes, although visible in the queries as changes of terms, cannot be regarded as a mere change of terminology. They comprise different, although related, concepts. An example of such a change is the change of "Peltier" to "cooling device", shown in Subsection 4.5.1. Such changes seem part of the second cause. Some changes observed are in the range between the second and third cause; the subtypes and supertypes of a concept often are quasi-synonyms, i.e., they are different terms for the same concept, but they may also be regarded as different concepts. Since concepts change essentially, it seems reasonable to state that the second cause is important. Hence, the first step of Section 4.1, i.e., obtaining an understanding of the knowledge lack, is important in the query formulation process.

This finding is in agreement with that of Shute and Smith (1993, p. 43), as obtained in studying the behaviour of an expert human search intermediary. They implicitly provide as conclusion: "Searching is not simply a matter of finding the best query to retrieve documents on an initially stated topic. A major part of the task is exploring the topic area so that the specific topic of interest can be identified."

The identification of concepts

From the audio recordings of the notes made in the search sessions, it appears that when obtaining an understanding of the user's knowledge lack, the user identifies (important) concepts. He does so in two ways. First, he describes concepts determining what the invention is about. The concepts so identified by description are usually very specific. The description relies on the structuring of knowledge in type hierarchies. The description of a concept is (in the current study) done by indicating its membership of a type hierarchy and by mentioning additional characteristic features (additional to the characteristics the concept already has on the basis of its position in the hierarchy).

Then, starting from these concepts, associations are made. The process of association serves to find alternative concepts to the invention, that have some identical essential characteristics, but also some differing characteristics. Association comprises the abstraction of characteristics, and subsequently often specialisation. For example, in a machine there may be nuts and bolts with the function of fixing two given parts mutually. Thus, a hierarchy of parts and a hierarchy of functions are distinguished; the hierarchies are crossconnected by relations[35]. The function of the parts given is abstracted, from "fixing part a and part b mutually" into "fixing parts mutually". Then, going back to the parts hierarchy, the types of parts having this function are identified, as alternatives to the original parts. This mechanism of association, viz. by abstraction in one type hierarchy and subsequent specialisation in another type hierarchy is known in artificial intelligence as heuristic match (Shapiro, 1992).

35 Chapter 5 elaborates on the constitution of knowledge representations and the presence of type hierarchies and their cross-connections. Chapter 6 provides illustrated examples thereof.

Query formulation pattern

In the simulated search tasks, most queries are formulated according to a similar pattern. This is observed in the tasks 1, 2, 5 and to a lesser degree in tasks 3 and 4. The identification of concepts as described above is part of the pattern. However, it is not a separate step but part of the identification of alternative terms (step 2). We regard the second step to serve both the identification of alternative designations for a concept and alternative concepts. The formulation pattern comprises the following steps, usually but not always in that order:

1. terms (hence concepts) are selected for the query, either based on the characteristics kept in mind, or from a previous query. The terms provide:
   – a general indication of the device or method in which the invention is embodied (by a main class/type name), and
   – specifying characteristics of the device or method, in terms of functions or technical measures (thus specifying a subclass/subtype),
2. for each term, alternatives are conceived of. The alternative terms serve as alternative designations (synonyms, quasi-synonyms, subclasses, and superclasses),
3. the alternative terms are connected by disjunction operators, and
4. the groups of terms resulting from (3) are connected by conjunction operators.

4.5.3 Conclusions and suggestions for future research

Conclusions

The theoretical study of query formulation recognises that a distinction should be made between the user's lack of knowledge and the user's understanding of this lack of knowledge.

The results of our empirical research are interpreted on the basis of the theory of Section 4.1. We draw three conclusions on patent searching, as performed in service of the writing of a patent application.

1. The user's understanding of his knowledge lack (i.e., his information need) changes noticeably, i.e., the first step mentioned in Section 4.1 (development of the understanding) plays an essential role.
2. Concepts are identified by their characteristics (direct or by association).
3. Queries are formulated according to a fixed procedure, in which the change of the information need and the identification methods mentioned above play a role.

All three conclusions are answers to our first research question. The above procedure was identified by a small-scaled study, for the specific task of patent searching. Therefore, the conclusions are tentative, and may hold only for the search tasks performed and the test person involved.

Regarding the fixed procedure, we are aware of the fact that it was not proven that it is an effective procedure. Yet, it may be expected to be effective, since it takes the first two conclusions into account, and it subdivides the process in a natural manner into two steps. The steps are first (1) deciding what you want in an exhaustive way, and then (2) expressing the latter in a manner suited for the matching process involved.

Chapter 6 formalises the fixed procedure. This activity results in the design of the query formulation tool Dipe-D. The latter tool structures the query formulation process, and thus supports the user in (1) the development of his understanding of the knowledge lack, and (2) the formulating of a query. In service of the development step, Dipe-D allows for the identification of concepts by description and - to some extent - by association.

Suggestions for future research

Although it was not proven by the above study, the conclusions may augment the theory of Section 4.1 in a general setting. In other words, other search tasks than patent searching may show the same results. Future research may explore this for all three conclusions.

A second interesting issue for future research is the simplification of the document-search task, as applied in Section 4.1: what effect has this simplification on the effectiveness of the search task, in particular on its precision. It is likely that a decrease of the precision results from the simplification, but it would also be interesting to find out the magnitude of such a decrease.

Chapter 5

Dipe-R: a representation language for query formulation

The chapter assesses our second research question: "*Is it possible to formulate a representation language for domain knowledge that effectively supports initial query-formulation processes?*" For this purpose, the chapter provides the design of a representation language for supporting query formulation (Dipe-R).

Dipe-R stands for Description and Identification by PropErties – Representation language. It is based on the idea that query formulation can be done by identifying query keys through the description and subsequent identification of the concepts they designate by their features (or properties, for the sake of the acronym). For supporting the description and identification, Dipe-R represents concepts, their features, and their hierarchical relationships, as well as query keys, and their relationships to the concepts identified. As such, Dipe-R supports query-formulation processes using these relationships, in particular the process of Chapter 6 (i.e., the two-step approach in our query-formulation tool Dipe-D, based on the findings of Chapter 4).

The chapter is structured as follows. It starts by formulating two requirements for Dipe-R (Section 5.1). Next, it discusses related research in knowledge representation, from which we learn lessons for Dipe-R (Section 5.2). Subsequently, the chapter elaborates on the constituents of knowledge to be represented by Dipe-R (Section 5.3). We then present Dipe-R's overall design (Section 5.4). In the remainder of the chapter (Sections 5.5 - 5.7), the design is worked out. Finally, Section 5.8 provides an evaluation of the chapter.

5.1 Two requirements for Dipe-R

According to the research question above, Dipe-R should effectively support initial query-formulation processes. In order to promote such support, we formulate two requirements for Dipe-R. Once these requirements are met, Dipe-R will be (theoretically) fit for supporting a wide range of query-formulation processes.

The first requirement is a basic requirement for representation languages, since it recognises elementary lessons of communication and knowledge (as elaborated in Chapter 2). The second

requirement focuses on query formulation, using lessons learnt in the Chapters 3 and 4. Figure 5.1 presents the requirements; they are elaborated below.

Requirement number	Requirement
1	**Proper expression of mental states and processes**
1a	Expressions simular to natural language
1b	Recognition of subjectivity and time-dependency of mental states and processes
1c	Recognition of origin of mental states and processes
2	**Identification of keys by the hierarchical relations and features of the concepts they designate**
2a	Relation of concepts to their sub/supertypes, and to their features
2b	Relation of concepts to terms, words, etc.

TABLE 5.1: *The two requirements for Dipe-R.*

Requirement 1

For the first requirement we use the lessons of Chapter 2, telling that knowledge is a property of mental states and processes, and that these are subjective and time-dependent. Moreover, we use the lesson that most mental states and processes have an origin, i.e., are based on something (e.g., an observation). Both properties are recognised in the first requirement, in order to obtain a proper understanding of expressions of mental states and processes:

1. Dipe-R should enable the appropriate expression of mental states and processes.

This is split into three sub-requirements:

1a. Dipe-R's expressions should be similar to natural language, in order to be clear to (almost) untrained users,

1b. Dipe-R should recognise the subjective and time-dependent nature of represented mental states and processes[36], and

1c. Dipe-R should recognise that most mental states and processes have an origin.

The sub-requirements are motivated as follows.

1a. Every person has command of at least one natural language. Natural language is the richest type of language we have. So, natural language is best fit for making clear expressions.

1b. A mental state or process is subjective and time-dependent, i.e., is someone's mental state or process, at a specific moment (or interval) in time. Recognising this, some misunderstandings can be avoided. An obvious solution is to store the name of the person whose mental state or process is represented and the moment he had that state or process.

36 As will become clear in Section 5.3, this includes mental states as well as their expression.

1c. Mental states and processes (including thoughts, see Section 5.3) often can be traced back to other mental states or mental processes or inputs, i.e., to their origin. For being able to disambiguate the meaning of expressed mental states and processes, and/or to assess their credibility, it is often helpful to have information on their origin (see Subsection 2.1.2).

Requirement 2

In Chapter 3 we learned that the formulation of the initial query is known to yield difficulties with respect to finding appropriate keys and concepts (and the proper use of (Boolean) operators). In Chapter 4 we learned an approach for query formulation in which: (a) concepts are identified by features and sub/supertypes (direct or by association), and (b) the identification of concepts is distinguished from the identification of keys for the query (since queries are formulated according to a fixed procedure in which this distinction is made). The effectiveness of this approach is still an open question, but – as we already mentioned in Chapter 4 – it seems an appealing approach. Therefore, on the basis of the above-mentioned lessons of Chapters 3 and 4, we formulate the second requirement:

2. Dipe-R should allow for the identification of keys by the hierarchical relations and features of the concepts they designate.

This is split into two subrequirements:

2a. Represented concepts should be related to their subtypes and supertypes, and their features, and

2b. Represented concepts should be related to keys (terms, words, etc.).

These subrequirements are motivated as follows. They are based on findings of Chapter 4: concepts of the information need are identified by hierarchical relationships and features, and keys are used for designating the concepts of the information need. In addition to these findings, it is generally known that hierarchical relationships play a large role in human thinking; human beings organise their knowledge by arranging concepts into type hierarchies (see Subsection 5.3.1). Moreover, the distinction of concepts from designations seems to correspond to the organisation of human knowledge: a single concept is often designated in several ways. This variety needs to be represented, in order to enable a query formulation tool to identify all query keys (for full-text indexed documents).

Dipe-R meets these two requirements. In Section 5.4, we provide its general design. First Section 5.2 explores the lessons learned from related research, and Section 5.3 provides theory for Dipe-R, by elaborating on knowledge.

5.2 Related research

The section explores related research in the field of knowledge representation. This field is too large to be reviewed in this chapter. We therefore make a selection; we only discuss representations of domain knowledge relevant to Information Retrieval. In the first place, these are representations common in Information Retrieval, such as (a) thesauri and (b) dictionaries (see also Chapter 3). Additionally, there are also representations from outside Information Retrieval that partially

meet our requirements: (c) semantic networks, (d) the KL-ONE family, and the products of two large-scale projects in knowledge representation, viz. (e) the UMLS Metathesaurus and (f) the CycL representation language. The distinction into the above four types (a) - (d) is adopted from literature; we realise that their boundaries are not sharply defined.

We do not discuss other kinds of representations, since they have a too different focus. For example, in rule-based representations (see, e.g., Buchanan and Shortliffe (1984)), and in case-based representations (see, e.g., the introduction by Aamodt and Plaza (1994)), the emphasis is on derivation. In logic, the emphasis is on (merely theoretical) formal properties, like consistency and completeness. We consider formal properties useful, but not of the highest priority for adequately expressing domain knowledge. The reason for this is that human beings often have inconsistent and incomplete knowledge, and still this knowledge seems useful. The alternative of creating a representation that satisfies these formal properties usually results in a representation in which hardly anything useful can be expressed. In connectionist representations the knowledge stored is essentially implicit, and they have – so far – not been focusing on the storage of explicit expressions of knowledge in a language easily understood by human beings – as desired for Dipe-R.

For each of the representations, we evaluate whether they meet our two requirements of Section 5.1, and see what other lessons we may learn from them. The outcome of the evaluation regarding the requirements is shown in Table 5.2, and elucidated in the text below.

a. Thesauri

Thesauri were already introduced in Section 3.3. The main elements of the standardised thesaurus are terms, and binary term relationships. The terms are names of concepts (as opposed to identifiers). The relationships should be loosely interpreted as instances of the sentence type "<term1> <relation> <term2>". For instance, with <term1> = "dog", <relation> = "narrower term", and <term2> = "mammal", the relationship becomes: "dog narrower term mammal".

The relationships in a standard thesaurus (i.e., a thesaurus according to ISO 2788) are of the kinds "narrower term" (NT), "broader term" (BT), "synonymous term" (SYN) and "related term" (RT) (ISO 2788, 1986). The RT relationship is a container for all kinds of relationships, of course excluding those contained in NT, BT, and SYN. It may involve whole and part, type and instance, or geographical inclusiveness (Jones, 1993). In practice, the RT relationship is used sparingly, only for connecting really closely related terms. In this manner, the user is enabled to find useful additional terms to a given term. This would not be the case if *all* terms somehow related would be connected by RT; too often it would yield useless additional terms to a given term. Thus, the art of building a good thesaurus lies partly in the proper choice of the RT relationships. According to Jones (1993), the synonymy relationship is used rather loosely, covering next to synonymous terms also variant spellings or word orderings, abbreviations, acronyms, and full forms, as well as scientific terms, popular terms, trade names, quasi-synonymous terms, and even antonyms (like "literacy" and "illiteracy"). Given the above, it will be clear that thesauri do not meet our requirements. Only the subrequirement 1a is partly met, since the expressions are more or less similar to natural language, and the subrequirement 2a is partly met, since type hierarchies are present. The subrequirement 2b is not met, since designations and concepts are not distinguished explicitly. However, via the relationships in a thesaurus it is sometimes possible to derive which designations belong to the same concepts.

Req.	Thesauri	Wordnet	EDR	Semantic networks	KL-ONE family	UMLS	CycL
1a	Partly met.	Met.	Met.	Partly met.	Partly met.	Partly met.	Met.
1b	Not met.	Not met.	Not met.	Not met.	Not met.	Not met.	Met.
1c	Not met.	Not met.	Not met.	Not met.	Partly met.	Not met.	Not met.
2a	Not met: Hierarchies present. Features: untypified relations	Partly met: Hierarchies present. Features: linguistic relation types.	Partly met: Hierarchies present. Features: a small number of binary relation types.	Met, but complex: Hierarchies present. Features: attributes, quantifiers, functions, predicates, etc.	Met, but complex: Hierarchies present. Features: attributes, functions, predicates, restriction rules, etc.	Met. Hierarchies present. Features: first & second order relations, contexts	Met.
2b	Not met. Terms serve as concept identifiers. Synonyms present.	Met. Terms and concepts are distinguished	Met. Variants of designations are distinguished from concepts.	Not met. Terms serve as concept identifiers.	Not met. Terms serve as concept identifiers.	Met. Variants of designations are distinguished from concepts.	Not met. No linguistic information.

TABLE 5.2: *Evaluation of knowledge representations with respect to our requirements.*

b. Dictionaries

A dictionary can be regarded a vocabulary supplemented by informal explanations of the terms in the vocabulary. It usually has a less formal nature than a thesaurus. However, dictionaries meant to be used on a computer system are, in contrast, given a semi-formal syntax. Examples of dictionaries to be used on a computer system are WordNet (Miller, 1995) and the dictionary resulting from the Electronic Dictionary Research (henceforth EDR) project (Yokoi, 1995).

WordNet (Miller, 1995) is a dictionary containing linguistic knowledge on English, among other things morphological and syntactical categories. It contains word forms, connected by binary relations of seven types: synonymy, antonymy, hyponymy, hypernymy, meronymy, troponymy, and entailment. Words are related to other words in terms of these types of relations. Synonymous terms are collected into sets, called "synsets". The relationships of words and the synsets offer a larger expressive power than a standard thesaurus. Its greatest contributions seems to lie in the linguistic variety offered; expressing features of concepts thus is supported only with respect to linguistic concepts. An ontology derived from WordNet is OntoSeek. It provides a knowledge representation language based on WordNet, extended with a few non-linguistic relations (such as "part of" and "has function"). It is developed for searching yellow pages on the www (Guarino, Masolo, and Vetere, 1999).

EDR (Yokoi, 1995) is a project meant for automated translation. It has resulted in a dictionary, comprising of a large number of Japanese and English words, terms, and sentences. The connecting element for the two languages is a collection of approximately 400,000 (represented) concepts, hierarchically classified. For each language, the words are related to (a representation of) concepts, to their grammatical characteristics, and to descriptions of meanings. The concepts and their relationships are regarded a semantic network. The authors claim that a small number (some 15) of (binary) relation types in the semantic network suffices. This finding is opposite to the finding in the CYC-project, discussed below. This contradiction may be explained by assuming that the required number of relation types depends on the preferred level of detail. We note that EDR offers grammatical characteristics of its content as well. This is useful in, e.g., translation tasks.

In WordNet and EDR, requirement 1a is partly met, but 1b and 1c are not met. Requirement 2b is well met, 2a is only partly met since concepts can only be identified by linguistic features, i.e., not by other features. In conclusion, dictionaries, in their present state, do not meet our requirements.

c. Semantic networks

We introduced semantic networks in Subsection 3.4.1. Semantic networks were initially developed for representing concepts underlying English words and relations among these concepts, as a model of human (associative) memory (see, e.g., Quillian, 1966, 1968).

Woods (1975) provided a seminal contribution to the theory of semantic networks. Besides giving an analysis of (then) common misunderstandings about their semantics, he proposed as elements of a semantic network:

– nodes, for representing objects in the world (e.g., "John" for a particular person),
– information about the objects represented by the nodes: attributes of the nodes, and values of the attributes (e.g., if there is a node "John", it may have an attribute height, with a value of five feet, as expressed in "John has an height of five feet."),
– functions and predicates (with n argument positions, $n \geq 1$) for expressing information about the objects represented by the nodes, along with information about the types of values of the arguments, and their possible or allowed value ranges,
– the existential and universal quantifiers,
– lambda-calculus.

The elements are based on the idea of representing objects in the world directly, not via representing mental states or processes (that may refer to objects in the world, see Chapter 2). Thus, our first requirement is not met. With respect to our second requirement, the distinction of attributes, quantifiers, functions, and predicates introduces semantic primitives that allow for the representation of features (sub-requirement 2a). Concepts and terms are not distinguished, thus sub-requirement 2b is not met. As an extra, lambda-calculus is proposed; this is an interesting item fit for representing very specialised issues, but not necessary for Dipe-R at the present (first) stage of its design.

d. The KL-ONE family

The explorations made by Woods were followed by the design of a semantic network: KL-ONE (Brachman and Schmolze, 1985). This network has later been used as the basis for a whole

family of representations. This again has led to Description Logics (DL)[37]. For example, two other members of this family are KRYPTON (Brachman, Fikes, and Levesque, 1983) and Loom (MacGregor and Bates, 1987).

KL-ONE is a framework for expressing assertions and descriptions. The main elements of KL-ONE are:
– representation of concepts, by name or description,
– two-placed predicates (although they are named differently), in particular the subtype relation type, and a relation type for relating parts to wholes,
– the functional roles of concepts (indicating what functional role a concept may have, e.g., sender of a message, or officer of a company),
– interrelations among the roles, imposing several kinds of restrictions on the binary relationships (e.g., what concepts are allowed to be related), and
– an inheritance mechanism.

In KL-ONE, no information is represented for dealing with the subjective and time-dependent nature of mental states and processes (requirement 1b). Some information is available on the origin of the represented mental states and processes (requirement 1c), at least for concepts: concept types can be 'defined' and 'primitive', depending on the way they emerged. In other words, for concepts (as a kind of mental states or processes) it is represented how they came about.

For concepts, 'types' and 'instances' are distinguished, and they are placed in type hierarchies. The two-placed predicates allow for the representation of features. Thus, requirement 2a is met. As with semantic networks, concepts and terms are not distinguished. Yet, it is possible to assign two designations for a single concept. This is realised by expressing a co-reference relation between two terms. As such, synonymy and spelling variations, etc. can be dealt with, but introducing more concepts than necessary. Thus, requirement 2b is partly met.

In KL-ONE, an inheritance mechanism is included. It derives inherited characteristics without allowing for exceptions. This kind of inheritance is considered inadequate (see, e.g., Touretzky, 1986), as it leads to contradictions. In successors of KL-ONE this problem with inheritance has obtained attention, the focus being on completeness, consistency, and tractability (see, e.g., Borgida *et al.* 1989; Borgida and Patel-Schneider, 1994). However, it seems that it is not recognised that derivation only yields potential mental states or processes, and that the ultimate decision whether a derivation is accepted depends on the user (regardless of consistency).

e. The UMLS Metathesaurus

The UMLS Metathesaurus (Lindberg, Humphreys, and McCray, 1993; Schuyler *et al.*, 1993 and UMLS, 1994) is a semantic network, compiled from a number of biomedical naming systems.

The general structure of UMLS consists of binary relationships among terms, words, and concept-identifiers. The expressions (for the binary relationships) of UMLS are similar to those of a thesaurus; they thus cannot always be easily understood. Hence, requirement 1a is not met. Also, UMLS provides no information on the source and origin of it expressions; it thus does not

37 Description Logics homepage: http://www.ida.liu.se/labs/iislab/people/patla/DL/index.html

meet requirements 1b and 1c, respectively. This is a disadvantage when different theories are co-existing, as occurs, e.g., in the medical area of dementia (Plugge, 1997). The simultaneous representation of such theories will not be clear to the users of UMLS.

UMLS expresses features by binary relations between concepts. The types of these relations are of a much larger variety than those of a standard thesaurus (some 30), including the subtype and designation relation types. Unfortunately, UMLS contains a large number of isolated groups of concepts, mainly consisting of the hierarchical relations types subtype, "child_of", "part_of", and "conceptual_part_of". Hence, although the structure of UMLS allows for the representation of features, they are hardly present in its present content. Thus, requirement 2a is supported by the structure of UMLS, but not by its present content.

Requirement 2b is met, since UMLS distinguishes between (represented) concepts and their designations (whereby each concept may have several designations and constituent parts thereof). This structure is well-utilised in the specific content of UMLS; it contains a large number of lexical variations of the designations (e.g., spelling variants).

As an extra to our requirements, the UMLS manual mentions the derivation mechanisms inheritance and transitivity as possible ways of derivation for UMLS. These derivation mechanisms must be created by the user though.

f. The CycL representation language

The CycL representation language was developed in the CYC-project, a large-scale effort to create a huge knowledge base of common sense knowledge[38] (Lenat and Guha, 1986; Lenat and Guha, 1990; Guha and Lenat, 1991; Lenat, Prakash, and Sheperd, 1991; Guha and Lenat, 1994; Lenat, 1995; Lenat, 1998). In CYC, common-sense knowledge is characterised as knowledge that typically everybody has, and often is considered too trivial for describing in encyclopaedias. An example of (factual) common-sense knowledge is "Water is a liquid under atmospheric pressure, between 1 and 100 degrees Centigrade." The knowledge is considered true 'by default', i.e., regarded true until there are strong indications otherwise. Next to factual knowledge, common sense knowledge can be procedural. Examples of procedural common-sense knowledge are methods for reasoning by inheritance and by analogy.

The CYC-project constitutes an attempt to reduce the well-known brittleness of expert systems (Lenat, Prakash, and Sheperd, 1991). However, the knowledge representation language (CycL) developed in the project has been first used in a different application area than those for expert systems, viz. for Information Retrieval. This area seems particularly suited for the default nature of the knowledge, since the performance of the retrieval is only affected slightly by (individual) errors (Lenat, 1995): a single error results in less, or less good, information for a human being; in an expert system it could cause a snowball-effect resulting in many other (silly) errors. Of course, the same may happen to a human being, but often he will see immediately that something has gone wrong.

38 The number of explicitly represented assertions is above a million (Guha and Lenat, 1994). Many more can be derived.

In the early days of the project, CycL contained the following elements:
– assertions; these were implemented as frames and slots for representing subclasses, superclasses, and several other types of binary relations. The frame refers to a first concept, the slot to a second. The slot type name indicates the relation type;
– numeric certainty factors, for indicating the degree of belief in each assertion.

Representations in this version of CycL were complete: all conclusions considered derivable from the facts and inference rules could indeed be derived. There were no rules or facts that could not be used in certain cases.

Brachman, Fikes, and Levesque (1983) formulate criticism on frame-slot approaches: the interpretation of this kind of representation is difficult. In particular, it is not clear whether a frame expresses knowledge about individual facts, or intensional definitions. On the basis of this criticism, the frame-and-slot language is replaced by a richer language, in which relations with higher arities than two can be expressed, as well as modals (e.g., "believing" and "wanting"), quantifiers, and second-order statements (Guha and Lenat, 1994). The new language essentially consists of first-order predicate calculus with a few second-order extensions. Moreover, the certainty factors are removed; it is considered too difficult to acquire values adequately representing the degree of belief in the accompanying fact. This holds in particular for the certainty factors resulting from inference. Certainty factors are kept only in cases where the numbers are clear and known. This seems an acceptable approach, since it is in line with our own everyday experiences in reasoning with uncertainty: human beings judge certainty in numbers only sometimes. We note that predicates having arities higher than two can be translated into predicates with arity two; see also Section 5.3.

The most recent element added to CycL is that of 'contexts' (Lenat, 1998). Contexts, in CycL being descriptions providing additional information to a communication, are considered necessary for an improved handling of the default nature of the represented knowledge, especially in derivation[39]. In effect, they speed up derivation by (heuristically) reducing the number of irrelevant derivations. Since the introduction of contexts, each assertion is considered true (by default, not by definition) *in a certain context*. Each context is specified as a collection of assumptions made simultaneously. Consistency is guaranteed within a context, not outside. In order to maintain overall consistency, the assertions in a context cannot be used directly in another context when derivations are being made. A procedure is developed for deciding whether an assertion can be used in a specific context for which it was not originally defined. The procedure also creates new contexts (some are only existing temporally), using second-order assertions for the specification of contexts. A consequence of using this (heuristic) procedure is

39 We note that, strictly, each communication has as context the whole universe. For a communication to be effective (truth plays no role, see Chapter 2), both sender and receiver need some common mental states. These can be obtained by additional communications from sender to receiver, or by the receiver assuming relevant mental states in the sender. A context thus can be regarded as mental states considered relevant (by someone, a some moment in time) for clarifying a communication. In CycL, a context can be regarded a description of such mental states.

that completeness is lost (being the price for speeding up derivation). In our view, this procedure is a psychologically plausible approach, i.e., similar to how persons maintain their opinions, beliefs, etc. Recently, the notion of context is further detailed: a context is specified as a region in a 12-dimensional space. It is recognised that other descriptions of contexts are possible, but these 12 dimensions are assumed to be sufficiently characterising (Lenat 1998). Unfortunately, the full details of CycL are not available to the public[40].

CycL almost meets our requirements. The requirement 1a is met by the predicates; each predicate is explained by a sentence. The requirement 1b is also met, since each predicate is provided with a name and date. The requirement 1c is not met; no information on the origin of the predicates is available. From the above it will be clear that the requirement 2a is met convincingly. Since no distinction seems to be made between terms and concepts, the requirement 2b is not met. As an extra, CycL offers derivation. The default-assumption is a step forward in comparison to the derivation in the earlier version of CycL, and contexts seem a powerful (heuristic) way of increasing derivation efficiency and reducing problems in derivation. However, we believe that another step could be made in reducing conflicts, at least in theory: by fulfilling our requirement 1c. Of course, it is an open question whether this requirement plays a large role, or that context information suffices in practice.

5.3 Knowledge revisited

In Chapter 2, we introduced the notions of mental state and mental process. The present section elaborates on types of mental states or processes relevant for the design of Dipe-R. The selection of these mental states or processes is based on what is useful for representing domain knowledge. In the creation of Dipe-R's design, as presented starting in Section 5.5, the mental states or processes discussed here seem useful for representing domain knowledge. The judgement of usefulness was obtained by trial and error, on an example domain of refrigeration (see Chapter 7) and on a small medical domain.

These mental states and processes mentioned usually exist for a longer time, in the sense that they can be remembered; this property is necessary for them to be represented in dipe-R. They are (1) concept, (2) relation, (3) quantity, and (4) thought. We note that each of the notions can be substituted in the semantic triangle (see Figure 2.2), as a replacement for thought/reference. We also note that these notions are used in various meanings in research literature, and some of these meanings may differ from ours.

The section also elaborates on mental activities, viz. the skills of (5) expressing mental states and processes and of (6) deriving new mental states and processes from old ones. Below, the six notions are discussed, in six corresponding subsections.

5.3.1 The notion of concept

We have already used the notion of concept, without elaborating on it. In the thesis, we regard concept a part of a mental state or process (see Section 2.1). Put more precisely, we use 'concept' in the sense of cognitive psychology (see, e.g., Neisser, 1987). Thus, we regard a

40 The reason for this being that CycL is commercially exploited, by CyCorp (located near Austin, Texas).

concept something in the human brain (instead of, e.g., something in language only). It is generally accepted that a human being makes distinctions in the world he conceives (observes or imagines); he isolates fragments of mental activity in such a manner that it enables him to remember (reproduce) them and to use them in thinking[41],[42]. We refer to such fragments when we use the term concept. As other psychological notions, concept is a vague notion (see Section 2.1).

Generic versus individual concepts

We call a concept a *generic concept* if it refers to a collection of elements (including concepts) having shared characteristics[43] (whereby a characteristic is a relation to one or more other concepts). We call a concept that is not a generic concept an *individual concept*. For each member of the collection of elements with shared characteristics there may be an individual concept. A generic concept is also called a *type*, and a member of the corresponding collection of concepts an *instance* (of the type).

Generic concepts can be interrelated by a subtype-relation. In everyday use, the subtype often shares one or more properties with the supertype[44] (we use this in Subsection 5.7.2). Individual concepts (instances) can be related to generic concepts (types) by an instance-relation. Structures created with generic concepts, individual concepts, and these two types of relations are called *type hierarchies*[45]. These structures are so often used in our thinking and communication, that they seem basic structures (Lakoff, 1987; Neisser, 1987).

Defined versus primitive concepts

For generic concepts, two types are distinguished, by their way of creation (see Quine, 1960; Brachman and Schmolze, 1985). The first type is a *primitive concept.* It is created when a person distinguishes a collection of concepts having something in common, without *explicitly* defining what. Optionally, he may define a few necessary conditions for concepts to be part of the generic concept, but not all the necessary conditions.

An important type of primitive concept is the *natural kind* (Brachman and Schmolze, 1985; Lakoff, 1987; Neisser, 1987). A natural kind is a primitive concept that has developed naturally, i.e., in the course of someone's life. "Dog" and "chair" are usually mentioned as examples. All members of a particular natural kind have characteristics in common, but it appears that membership cannot be simply expressed in a simple collection of necessary and sufficient conditions. Any attempt to obtain such a collection either yields exceptions that do not fit the collection, or do not include all the instances that most persons would regard members of the kind (Wittgenstein, 1953). Of course, each individual person has a different understanding of a

41 It is not said that each concept is experienced consciously.

42 The notion of thinking is elaborated in Subsection 5.3.6.

43 Oxford's Dictionary (Concise, 1995) defines type as "a class of things or persons having common characteristics".

44 The relationship between supertype and subtype is more complicated than we can treat here, see also Brachman (1983) and Brachman (1985).

45 Alternatively, they are called class hierarchies; types then are called classes.

specific kind; still it is commonly believed that for many such types essentially the same typical characteristics are recognised by (almost) everybody.

The second type is called a *defined concept*. It is created by the recognition of a set of (explicitly recognised) conditions that are each necessary and that together are sufficient. Defined concepts play an important role in mathematics and in the mathematical modelling in physical sciences (e.g., physics, chemistry). Mathematical concepts need to have precisely known characteristics to enable theories with sufficient precision. Natural kinds are often too vague for that purpose.

5.3.2 The notion of relation

In this thesis, a relation is characterised as a concept consisting of a connection of two or more other concepts[46]. The relation serves to represent a connection, or relationship, of two or more things in a world conceived. We do not know the nature of the connection of concepts at this point (and neither do we know the precise nature of the concepts connected, as mentioned above). In (Western) natural languages, relations in the above sense can be expressed by verb phrases. For instance, "Peter eats an apple" expresses a relation between Peter and an apple.

Binary relations

In Section 5.5, we will choose to represent binary relations in Dipe-R. Some knowledge contains binary relations. Some other knowledge contains relations among more than two concepts, but can be re-conceived as (other) relations between precisely two concepts, viz. by capturing a relation between two concepts into one concept and repeating this until only binary relations are left. It is not clear how easily the concepts that result from this process are remembered in comparison with the original (not-binary) concept.

However, there is an indication that they are easily remembered. This is the fact that many natural-language expressions seem to reflect binary relations (Woods, 1975): often, a relation is expressed as a verb phrase, connecting two noun phrases (each designating a concept). For example, "This book *is contained in* a cardboard box." The re-conception of relations as described above is also reflected in language: relations of higher arities, e.g., "John sold Mary a book"[47] can also be expressed as binary relations. This is done by assembling a relation and its accompanying concepts into a new, single concept. E.g., "the ship propelled by steam" can be regarded as a single concept, and designated alternatively by the name steamship.

Relation types

It seems plausible to assume that human beings distinguish types of relations[48]. For example, in a sentence the relation type is designated (by a verb phrase, as mentioned above), and not the individual relation. It seems plausible to assume that many relation types are learned in everyday life, by means of induction; they thus are natural concepts. For relations types that have been induced,

---

46 This implies that a relation, in our sense, comprises at least two concepts; this as opposed to propositional logic, in which a relation may also have only one concept (Copi, 1982).

47 The example is taken from Woods (1975).

48 This is based on the same type of observations and introspection on which we based the existence of concepts

it is usually also induced that their instances (i.e., relations) always relate two (generic) concepts of two specific types. We call the latter two concepts "roles" of the relation type (cf. Nijssen, 1993).

As an alternative to being induced, relation types may be defined (or have a different type of source); they then are defined concepts. The definition may include the two generic concepts serving as roles, but may also be concerned with the meaning of the relation type (whereafter it may be learned by induction *what* concepts can be related, i.e., what the roles are). Moreover, the definition may include information telling for which kinds of thoughts the relation type is meant (see Subsection 5.3.4).

5.3.3 The notion of quantity

A specific type of concept is that for quantities. Quantities are used in measurements, and are expressed on a scale. For instance, "Peter eats one apple" expresses a quantity, indicating the number of apples (one piece). Another instance, "Peter eats 100 grams of chocolate" also expresses a quantity, indicating the weight of the chocolate (100 grams). In these examples, a quantity (one respectively 100) occurs in combination with another concept, indicating the scale: piece (on an ordinal scale) and gram (on a ratio scale), respectively.

We only discuss a specific way of using quantity, relevant for Dipe-R. For this purpose, we inspect the expression "An instance of x R an instance of y". Here, x and y stand for concepts, R stands for a relation type. For example, "An instance of human being eats an instance of apple". This expresses a relation between two objects, in which the objects are not identified. If many such relations between pairs of concepts need to be expressed, the same type of expression can be used a number of times. Alternatively a single expression may be used, such as "Many instances of x R an instance of y". "Many" is regarded a way of indicating a quantity on an ordinal scale (i.e., a scale on which only a binary ordering exists among the quantities). Other quantities are, for example, "some" (i.e., at least one), "few", "several", and of course the numbers. Such a single expression can be regarded as an expression of a single mental state or process summarising a number of other mental states or processes.

5.3.4 The notion of thought

We use the notion of thought as a mental state or process about something conceived in a world: an opinion, idea, belief, etc., as opposed to a mere feeling, mood, or emotion (cf. Ryle, 1949). Of course, a thought may result from emotion, feeling, etc. An example of an expressed thought is "Rodham-Clinton is senator of the state New York".

As visible from the example, a thought is composed of concepts. In the example above, the thought comprises a relation and two concepts (one of which can be further decomposed into two concepts, viz. 'senator' and 'the state New York'). In some thoughts, quantities are also present. For instance, "*Many* journalists write about the senator." Other elements may be present in thoughts as well, but we will not elaborate on them since Dipe-R will not include them. Moreover, thoughts may comprise relations among more than two concepts, and also higher order relations (cf. Copi, 1982).

5.3.5 Expression of thoughts

Most people are able to express many of their thoughts in natural language. It is a matter of ongoing research how they do this, and how the thoughts and their expression interact (e.g., to

what degree they can be regarded to exist separately). On the basis of introspection and of observation of persons acting in daily life, it seems reasonable to assume that there exist mental states and processes with some stability, and that a single mental state or process often can be expressed in different expressions (e.g., depending on the specific context). Moreover, since expression is a mental process, it may also differ among persons and over time (see Chapter 2). Yet, due to (intersubjective) language conventions many similarities can be recognised among persons and over time. The latter is used for the expression by Dipe-R of pre-represented thoughts.

5.3.6 Reasoning and derivation

Much has been written about reasoning and derivation. Here we adapt views of Wittgenstein (1953) and Ryle (1949) concerning these two issues, expressed in the vocabulary of this thesis.

The mental states and processes of a person change with time: one mental state or process succeeds another. We focus on a subset of mental states and processes, viz. thoughts (see Subsection 5.3.4). Sometimes, the succession of thoughts is more or less at random. However, many persons develop in their lives the skills of having thoughts succeeding each other in an orderly way. It depends on the person what he regards orderly. Typically, a person regards only a few general types of successions as allowed for; usually these are determined by conventions (intersubjectively). Such allowed successions are called "rational" thinking. There is a wide consensus that reasoning considered rational consists of the succession of thoughts according to one of three general types: deduction, reduction, and induction (Eekels and Roozenburg, 1979). Various subtypes are recognised. Processes of reasoning, either or not considered rational, can be communicated. Toulmin (1959) describes how the succession of thoughts can be communicated and motivated.

Some kinds of reasoning can be described formally. The best-known example of this is numerical calculation. The formalised process can be implemented in a computer[49]. Then it may be used for simulating, by derivation, how a person's thoughts could, or should, proceed. Of course, the reasoning is only simulated; how the thoughts really evolve is not certain. There are at least two reasons for the uncertainty. First, a person may follow a different line of reasoning, or not reason at all (e.g., he may have mental states and processes that are not thoughts). Second, a person may not accept the conventions assumed in the formalisation. Hence, only potential thoughts (useful for making predictions of thoughts) are obtained by derivation. Nevertheless, for certain kinds of reasoning the predictability is quite good, e.g., the transitivity of relations, and the inheritance of properties (as long as no conflicts arise). This predictability is used for derivation in Dipe-R (see Section 5.7).

5.4 Overall design of Dipe-R

The design of Dipe-R is directly in service of meeting the two requirements of Section 5.1. The present section provides the overall design. The subsequent Sections 5.5 - 5.7 work out the details.

---

49 As a symbol manipulating machine, see Chapter 2.

Requirement 1

Dipe-R allows for the representation of thoughts (i.e., mental states or processes), and the representation of the mental activity of expressing thoughts in English sentences. We distinguish two kinds of represented thoughts in Dipe-R. A thought is not represented by a sentence, but by a different kind of representation, and a sentence is created when asked. This allows for different kinds of expressions for each thought. However, for each of the two thought types Dipe-R has essentially one sentence type; there is only variation in the designation of concepts. This variation serves in particular the provision of query keys. Thus, there are two sentence types in Dipe-R.

We assume that a language has syntax, semantics, and pragmatics, cf. Winograd (1983). The syntax of the sentences in Dipe-R is sometimes slightly incorrect, since it ignores the conjugations for differentiating between singular and plural form. The reason for this incorrectness is that it allows for a very simple expression procedure, without affecting the clearness of the expressions. The semantics of the expressions is identical to that of the corresponding English sentence. The two sentence types are chosen in such a way that little context information seems necessary. Probably, they can be understood once the user knows that there are only these two sentence types in the knowledge representation. Thus, it is, e.g., not necessary to know all the instances of the sentences in a representation (i.e., the whole content of the representation).

As a general remark, we mention that the content of knowledge representations in Dipe-R is created through knowledge acquisition by means of communication. Thus, the expressions are known first. Then, they are translated to a representation of thoughts in Dipe-R. Later, when the representation is consulted by a program, the represented thoughts are expressed again, now by Dipe-R.

The above properties of Dipe-R meet requirement 1a. For meeting requirement 1b, thoughts are represented as being subjective. Each represented thought is provided with an identification of the source, the source being a person or a document[50], and with a representation of the moment in time that the thought existed (according to the person whose thought it is).

Requirement 1c is met by providing information on the origin of each represented thought. The origins are instances taken from the classification of Section 2.4. The information mentioned for sub-requirements 1b and 1c is referred to as "source information" in the remainder of the chapter.

Requirement 2

As mentioned above, Dipe-R has two types of represented thoughts. They are distinguished by their constitutive elements. Both types represent binary relations. One type also includes quantities. By these two types, type hierarchies as well as features of concepts can be represented. Moreover, the relationships among concepts, designations, and words are represented. The latter is realised by recognising that designations and words, i.e., symbols, are also concepts. In the same manner as other concepts, the binary relations of the two thought types of Dipe-R thus can relate symbols to concepts (including symbols). Only when a represented thought containing a symbol concept is presented to a user, the symbol concept is treated differently from other concepts. With the above structure, the requirements 2a and 2b are met.

---

50 The direct source is always a person, the indirect source may also be a document; see Section 5.6.

Dipe-R does not represent relations among three or more concepts. Neither does it use second order relations, as found in CycL. They are not necessary and would make the implementation more difficult. It is also not necessary to distinguish attributes, functions, predicates, roles, and restrictions (as introduced in semantic networks, in KL-ONE, and elsewhere, see, e.g., Braspenning, 1989; Guarino, 1992). As a working hypothesis of Dipe-R, we assume that binary relations alone suffice (see Subsection 5.3.2).

As an additional measure for meeting requirement 2, Dipe-R contains derivation rules and a derivation procedure. Derivation is also present in some of the knowledge representations discussed in Section 5.2. In Dipe-R, the rules and procedure are domain-independent, in line with recent developments in knowledge representation; they thus can be used in each domain represented. The rules have as parameters represented thoughts of Dipe-R, including the accompanying source information. However, in the present version of Dipe-R the source information is only used to distinguish derivation from original thoughts, and not for more sophisticated reasoning such as resolving conflicts between represented thoughts (or expressions thereof).

Dipe-R is worked out in Sections 5.5 - 5.7. Section 5.5 elaborates on the representation of thoughts and their expression. Section 5.6 elaborates on source information, and Section 5.7 on derivation.

5.5 Thoughts and their expression in Dipe-R

Dipe-R represents thoughts and their expression, in such a manner that both requirements are met. Dipe-R distinguishes thoughts from their expression; this is done for being able to create more than one expression of a thought. We recognise that a different expression may also imply a different thought, as discussed in Section 5.3. Subsection 5.5.1 presents what types of thoughts Dipe-R represents, Subsection 5.5.2 presents thoughts pre-represented in Dipe-R, and Subsection 5.5.3 elaborates on how Dipe-R expresses represented thoughts.

5.5.1 The representation of thoughts in Dipe-R

We first introduce the two kinds of thoughts that Dipe-R represents informally, and then specify these formally.

The two kinds of thoughts represented

For meeting requirement 2, we want to represent thoughts that relate concepts to their sub- and supertypes, and to their features (requirement 2a). This includes relating concepts to designations and words, etc. (requirement 2b).

By trial and error, we discovered that one solution to realise this consists of two specific types of represented thoughts. These were found by experimenting on a sample domain of refrigeration (see Chapter 7) and on a small medical domain. Both types include binary relations. This choice coincides with the approaches of UMLS (1994) and the standard thesaurus (ISO 2788, 1986), and we expect that it allows for the representation of a useful amount of knowledge. The main difference between the two types of represented thoughts is that one of them is meant for referents that are concepts themselves. The other type of represented thought is meant for referents that are not concepts.

In Dipe-R, we regard represented thoughts of these two kinds as constituted by represented concepts. We mention as specific subtypes of represented concepts in Dipe-R: represented relations, represented designation types, and represented quantities.

1. A thought of the first type refers to two concepts, relates them, and identifies both concepts. This type of thought allows for the creation of type hierarchies, and the representation of features. Moreover, it allows for the relation of designations to concepts. It can be expressed in a statement type of the following structure:

   "c1 r c2".

Here, c1 and c2 are variables for concepts, and r is a variable for a relation type. An example statement of this type is: "(The concept) dog is a subtype of (the concept) animal".

2. A thought of the second type refers to one or more pairs of referents, and relates them without precisely identifying them (e.g., *a* person, instead of the person *Peter*). It allows for the representation of features, both one-by-one and several simultaneously (thus providing a high expressiveness). A thought of the second type can be expressed in a statement type of the following structure:

   "q1 instance(s) of c1 r an instance of c2".

Here, q1 is a variable for a quantity-indication, and c1 and c2 are variables for concepts, and r is a variable for a relation type. The quantities q1 and "an" enable the reference to several pairs of referents simultaneously. The words "instance" serve to increase the clarity of the statement and to reduce the amount of context information needed for a proper understanding. Two example statements of this type are: "Many instances of dog bite an instance of leg", and "An instance of dog eats an instance of amount of food". It is not specified which instance precisely is meant.

We note that c1 and c2 may be any concept, with a countable designation as well as an uncountable designation[51]. Expression in natural language yields expressions differing somewhat, by using a singular and plural form. As already mentioned in Section 5.4, Dipe-R does not make this distinction.

We note that some thoughts can be represented by each of the above types. For instance, "treatment a treats disease b" (wherein a and b are names) can also be phrased as "all instances of treatment type a treat an instance of disease type b". In such cases, the person creating a representation has to decide; he should use type 1 for concepts as referents, and type 2 for other things as referents. In the future, Dipe-R may obtain derivation rules for recognising and "translating" expressions of both types. Because of this possibility to use both represented thought types, expressions in Dipe-R are context-dependent, but the context knowledge needed

51 "Sugar" is an example of an uncountable designation; "dog" is an example of a countable designation (one dog, two dogs, …).

for a proper understanding is limited to knowledge of the two thought types and their expressions (as opposed to, e.g., the whole content of the knowledge representation).

As mentioned in Section 5.4, each represented thought (of the two kinds above) is paired with additional information, providing the name of the person whose thought it is, when he had the thought, and how he obtained the thought. We name this information "source information". Thus, a thought is treated as subjective and time-dependent, and being caused by something (see also Chapter 2). Only thoughts are provided with such information, i.e., not their constituting elements. Their elements are, in contrast, treated as intersubjective. For source information: see Section 5.6.

Specification of Dipe-R, regarding thoughts

We specify Dipe-R in Backus-Naur Form notation (Naur *et al.*, 1963), in such a manner that it can be implemented in PROLOG straightforwardly. Dipe-R's specification is divided over the Sections 5.5 – 5.7. It is summarised in Appendix A. Here, we specify how it represents thoughts, and summarise what thoughts are pre-represented in each implementation of Dipe-R.

<represented_thought> ::=
th(<represented_thought_id>,<content>, <source_information>).

<content> ::=
[tt_1, <represented_relation_type_id>, <represented_concept_id$_1$>,
<represented_concept_id$_2$>] |
[tt_2, <represented_relation_type_id>, <represented_quantity_id>,
<represented_concept_id$_1$>, <represented_concept_id$_2$>]

<represented_thought_id> ::= integer number (1,2,3,…).
<represented_concept_id> ::= integer number (1,2,3,…).
<represented_relation_type_id> ::= integer number (1,2,3,…).
<represented_quantity_id> ::= integer number (1,2,3,…).

<source_information> is discussed in the Section 5.6.

The thought type identifiers "tt_1" and "tt_2" correspond to the first and second thought type introduced in the text above, respectively. The thought_id serves to distinguish different represented thoughts having identical contents. The position of an identifier in <content> determines its type; thus, it is allowed for identifiers of the four types mentioned to have overlapping numbers.

We provide an example of <content>. For better readability, the concept identifiers (numbers) are replaced by designations. It now reads:

[tt_2, comprise, all, Brayton cycle, adiabatic compression of gas]

This is expressed by Dipe-R as "all instances of Brayton cycle comprise an instance of adiabatic compression of gas".

5.5.2 Dipe-R's pre-represented thoughts:

Several thoughts are pre-represented in each implementation of Dipe-R. They are necessary in all implementations of Dipe-R. The pre-represented thoughts are enumerated in Appendix A. They represent:
– The existence of specific relation types,
– The roles of the relation types, and
– In which thought type(s) each of the relation types is allowed to be used.

The relation types serve to provide structure among concepts (subtype, transitivity), and to provide query keys: has_designation, etc. The roles serve as restrictions for relation types, see Subsection 5.3.2. It is represented in which thought type(s) a relation type is allowed to be used, as a restriction – in analogy to roles. Of course, a relation type that relates only concepts is allowed to be used in the first thought type, and a relation type that only relates objects is allowed to be used in the second thought type. Many relation types are allowed to be used in both thought types.

The subtype relation type is introduced in order to be able to create type hierarchies. The subtype relationship can be used to derive or guess features of the subtype from the supertype, vice versa; see, e.g., Brachman (1983). We discuss this further in Section 5.7.

5.5.3 Expression by DIPE-R

Dipe-R has a fairly simple expression procedure. It produces, from represented thoughts, propositional statements that are sentences in English. The procedure has two variants, a basic and an extended variant. The extended variant shows the roles of the concepts in a relation, whereas the basic variant does not show these.

Since there is only one procedure for expressing represented thoughts, Dipe-R cannot simulate the different ways different persons would express their thoughts. Thus, the procedure and some of the information it uses (viz. the – partly pre-represented – thoughts for relation types, roles, and in which thought type each relation type is allowed) are regarded as intersubjective. The expression procedure (and the explicit information it uses, see below) therefore is not provided with source information.

Basic variant

In the basic variant, a procedure E performs the expression:

<statement> ::= E(represented_thought)

For each represented thought, E creates a statement starting from a sentence type template. The templates are those of the two sentence types introduced in Section 5.5.1. For thought type 1 and thought type 2, they are defined as follows:

<sentence_type_1> ::= "<2> <1> <3>."
<sentence_type_2> ::= "<2> instance(s) of <3> <1> an instance of <4>."

The procedure E is informally described as:
– find the sentence-template for the thought type of the represented thought;
– assemble an instance of a sentence, by parsing the template and substituting the variables in

the slots. Each variable-slot in the template is either substituted by the identifier of the variable, or by a designation.

The parser of the template recognises the dot symbol as the end sign of the sentence. It also recognises the slots for variables, by the "<" and ">" symbols, and reads the number of the variable.

In order to determine whether a variable identifier or a designation must be used, the parser checks whether the variable is a quantity concept, a symbol concept, a relation concept, or another, different concept. It does so by checking the thought type definition: by the position of the variable in the thought, its type can be known from the thought type definition; at least, quantities and relation types are distinguished from each other and from symbol concepts as well as from different concepts. Next, symbol concepts and different concepts need to be distinguished; the relevant information for the distinction is obtained from the role information of the relation type. In the case of quantities and symbol concepts: the (literal) identifier, i.e., the designation, is used. In the other cases, a designation is found and used (through a different represented thought, relating a concept to a designation, by the "has_des" relation type).

Extended variant

This procedure is almost identical to the basic variant, the difference being that the relation type is specified into more detail: the two roles associated with the relation type are mentioned[52]. For example: "The *compressor* P-x139 contains the *part* 12345."

By adding roles, the relation type is designated more precisely than by using a verb only. Still the roles are very general, providing only information on the concepts allowed for by the relation type. More specific information on the concepts then has to be obtained from other statements in the representation. This approach of relation types is based on ideas developed by Nijssen and Halpin (Nijssen and Halpin, 1989; Nijssen, 1993); they applied the ideas in the information modelling language NIAM. From Nijssen, we adopt the name "roles" for the generic concepts defining a relation type.

In the modelling languages NIAM and DEMO, sentences contain even more information than the roles of the relations: the types of the concept designations are mentioned too. For example: "The compressor with the *name* P-x139 contains the part with the *number* 12345." Below, we adopt the approach of mentioning roles. We do not consider it necessary to mention the types of the concept designations too; in most instances they are obvious.

5.6 Source information

In line with the general design of Section 5.4, Dipe-R stores source information for each represented thought. It stores information on (1) the person or the document where the thought is taken from, and (2) a moment in time the thought existed, and (2) the origin of the thought. Source information thus meets the subrequirements 1b and 1c.

52 The roles may be obtained from different origins, as discussed in Section 2.4 (e.g., by definition, or by induction).

We decide to represent source information along with the content of a represented thought. This allows for a simple implementation (not needing second order expressions or other difficulties), and still allows for the source information to be used. However, we keep in mind that in future extensions of Dipe-R source information may be represented in a different manner, allowing source information to be treated in the same manner as content information. For example, it may then be identified by the same procedure that also identifies concepts of the content by their features. The source information of Dipe-R comprises:

1. The name of the person holding the thought, or the URL of the document from which the represented thought was drawn,
2. A time indication of a moment the person held the thought,
3. The origin type,
4. Additional information.

Source information is defined as follows:

<source_information> ::= <source_id>, <time_stamp>, [<origin_type>],
[<additional_source_information>]

<source_id> ::= <person_name> | <URL>
<person_name> ::= name of the person whose thought it is.
<URL> ::= universal resource locator (on the www).
<time_stamp> ::= <year, month, day>
<year> ::= year in four digits.
<month> ::= month in two digits.
<day> ::= day in two digits.
<origin_type> ::= **original, definition** | **original, observation** |
original, communication | **original, derivation,**
induction | **original, derivation, reduction** | **original,**
derivation, deduction | **original, derivation, mix** |
original, other | **derived**
<additional_source_information> ::= <communication_path> |
<derivation_path>
<communication_path> ::= <source_id> |
<source_id>, <communication_path>
<derivation_path> ::=<represented_thought_id> |
<represented_thought_id>, <derivation_path>

The name of the person and the time indication[53] are meant to allow the user to obtain additional information, when needed. Such additional information may, for example, be used to assess the credibility of the thought. Our reason for storing a document URL instead of the

53 Of course, the time indication above assumes that a single time scale can be applied. For the purposes of Dipe-R foreseen, i.e., non-relativistic, this suffices.

name of a person is that we expect that many of the represented thoughts may be obtained from documents; nowadays, much information is available from digital documents, as stated in Chapter 1. It seems useful to be able to retrieve a document from which a thought was taken. Of course, each represented thought is taken from a person. However, that person may have adopted a thought from a document. Furthermore, the document may mention the person who held the thought in the document. This communication path (person, document, person) can also be represented (see below).

The source types are subdivided into original and derived. The source type is called original if the corresponding represented thought is actually held by someone. The source type is called derived if the corresponding represented thought is obtained by formal derivation by the derivation procedure of Dipe-R (it thus is a potential thought).

The source type original is further subdivided, almost according to the source types recognised in Subsection 2.1.2, and mentioned by Audi (1998). We have streamlined the latter source types, in order to enable that for communication and derivation the source can be further specified (by a path). We distinguish the following source types: (1) definition, (2) observation, (3) communication, (4) derivation, and (5) other. Moreover, derivation is further subdivided into (1) induction, (2) reduction, (3) deduction, and (4) mix.

We note that a represented thought with the source type derivation, i.e., a represented *potential* thought, may be agreed upon by a user of Dipe-R. It then can subsequently be represented anew, now as an original (i.e., realised, not derived[54]) thought of that user. Hence, the thoughts having identical contents are then represented twice.

Additional information is provided only if the source type is "original, communication" or "derived". For communication, only the sources in the communication path are stored, not the thoughts. It is meant for being able to represent that a thought is not obtained at the first hand. Since the additional information can be defined recursively, we allow for storing arbitrarily long communication paths. For derived thoughts the entire derivation path is stored, consisting of all thoughts used including the derivation rules.

5.7 Derivation

As mentioned in Section 5.4, Dipe-R contains a derivation procedure. We explained in Section 5.3 that this allows for the representation of potential thoughts. By deriving obvious potential thoughts, that are not explicitly represented, less labour for representation is required. However, the advantage is at the cost of some certainty: the potential thoughts derived may be not accepted by the origins. As explained in Section 5.3, Dipe-R only provides derivations that are very likely to be accepted by the origins and that seem to provide useful information. Hence, the disadvantage plays no practical role in Dipe-R. Below, we provide the derivation procedure and the derivation rules. We first present how Dipe-R supports derivation and introduce the procedure (Subsection 5.7.1), and then present the derivation rules (Subsection 5.7.2).

---

54 In line with this view, the distinction into original vs. derived thought (inspired by the notions of 'original fact' and 'derived fact' of Dietz, 1994) could be replaced by the distinction realised vs. potential; the latter distinction seems more characteristic.

5.7.1 Derivation in Dipe-R

For derivation, both a representation formalism and a derivation procedure are needed. The two predominant types of representation formalisms are rule-based and case-based. In Dipe-R, we support a rule-based formalism. As derivation procedure, we use the standard PROLOG formalism. Below, we motivate the two choices for rule-bases and PROLOG.

Choice of the representation formalism

According to our view on derivation, both rule-based and case-based formalisms have in common that they relate representations of thoughts to representations of potential thoughts.

In rule-based formalisms (see, e.g., Buchanan and Shortliffe, 1984), both individual thoughts and collections of thoughts may be interrelated. In case-based formalisms (see, e.g., Aamodt and Plaza, 1994), cases (each having the same description structure) are related. A case represents a collection of thoughts, and a case type is a collection of thought types. Thus, case-based formalisms relate specific collections of thoughts. As such, they may be viewed as a subtype of rule-based formalisms having incomplete cases as conditions and complete cases as conclusions. The latter view is not common, perhaps because the way of relating cases is usually not expressed in rules that resemble rules of the rule-based formalism (although they can be). This view makes it obvious that, strictly spoken, each choice is a choice for a rule-based formalism.

Rule-based formalisms are more general than case-based formalisms. Moreover, in modern approaches the former are formulated in such a manner that the procedure and the rules are domain-independent, and only the thoughts they operate on are domain-dependent. Thus, the procedure and the rules can be reused in many domains (see, e.g., Van Heijst, Schreiber, and Wielinga, 1997; Van Hyfte *et al.,* 1999).

In Dipe-R, derivation is pre-dominantly meant for using the transitivity of the relations, and for the inheritance of features in type hierarchies. These types of derivation relate individual thoughts, and thus are best represented by a general rule-based formalism (i.e., not case-based). This explains our choice.

Choice of the derivation procedure

A derivation procedure preferably simulates (potential) reasoning, in order to predict thoughts acceptable for human users. Our research has Information Retrieval as focus. However, the development of such realistic derivation procedures is far too large an enterprise in the scope of this thesis. Instead, we adopt an existing procedure, viz. the procedure implemented in most versions of PROLOG. It is a fairly straightforward procedure, performing resolution as LDF backward chaining (Bratko, 1990).

We adopt it, with one remark. The standard derivation procedure of PROLOG is often interpreted as following the closed-world assumption (Rowe, 1988). In those interpretations, the information base of PROLOG is interpreted as consisting of facts. A statement not stored in the PROLOG information base, and not derivable from it by the PROLOG derivation procedure, is then interpreted to express something false. In other words, the description of the subject as available in the information base and by the derivation mechanism, is assumed complete and consistent; something is known as true or false, and there is nothing not known.

In Dipe-R, we use the following interpretation: the description of the subject area is assumed incomplete. For real-world domains, this seems a more realistic assumption than completeness.

Hence, if a thought is not represented, it only means that no information on it is available, but it does not mean that it cannot be a thought. The same holds for a (potential) thought that cannot be derived by PROLOG from the information base: it may still be or become the representation of an existing thought. We therefore regard it unknown. Thus, we interpret the NOT-operator in PROLOG as "not known", next to the confirmation. Of course, in PROLOG this takes away the possibility to represent that something is false.

In Dipe-R, a derived thought is assigned the source type "derived". As such, it is distinguished from original thought types. Once a potential thought is derived, it can be presented to a user, who may then agree or disagree, as already mentioned in Section 5.6.

We note that the derivation rules and the derivation procedure represent thoughts[55], and thus could be treated as subjective and time-dependent. However, Dipe-R is given only the present set of rules and the present procedure; these seem acceptable for typical users.

5.7.2 Pre-represented derivation rules

Below, we present two types of pre-represented derivation rules: (1) for the transitivity property of relations, and (2) for the inheritance of features.

The rules do not use source information in their conditions, although we mentioned (in Section 5.6) that source information plays an important role in reasoning. As stated above, it is, for instance, used for assessing the credibility of statements (e.g., in a numeric certainty value, or a in certainty value on an ordinal scale). Our reason for leaving out source information from the conditions stems from our decision to exclude ways of expressing the credibility of statements from (the present version of) Dipe-R (see Section 5.4). However, the conclusions of the rules do yield source information, that may be used for assessing the credibility of derived thoughts.

The notational convention is given in Backus-Naur Form, in which a rule is represented as a PROLOG-rule (cf. Bratko, 1990).

<derivation_rule> ::= <represented_act> :- <represented_acts>.

<represented_acts> ::= <represented_act> |
<represented_act>, <represented_acts>

<represented_act> ::= <represented_thought> | <get_th_id>

<get_th_id> ::= assignment function for identifiers of represented thoughts.

<represented_thought> was already defined in Section 5.5.

The symbol "_" can be used on the position of a variable, indicating that an arbitrary value is allowed for (as in PROLOG).

A name starting with a capital indicates a variable, a name starting with a lower case letter indicates a constant (as in PROLOG).

55 Viz., thoughts of the thesis author.

In the rules below, we replace the identifiers (numbers) of the relation types and different concepts (i.e., concepts being not symbols, not quantities, and not relation types) by their designations; this serves to improve readability.

1. Rules for the transitivity property of relations

Some types of relations are regarded transitive (Doets and Troelstra, 1983). Transitivity means that if someone has two beliefs, "a r b" and "b r c", with a, b designating concepts, r designating a relation type, and the relation type "r" transitive, that person is likely to accept "a r c" as well[56]. Three examples of transitive relation types are (1) the subtype relation type, (2) the relation type for causality, and (3) the relation type of containment (often indicated as "part of").

Below, we formulate two transitivity rules, one for the first thought type of Dipe-R, and one for the second thought type. The rule for the first thought type is the formal version of the rule introduced above (and has as represented_thought_id the number 2558).

```
th(Id1, [tt_1,R,X,Z], _, _, [derived], [Id2,Id3,Id4,2558]) :-
   th(Id2, [tt_2, has_characteristic, all, R, an transitivity], _, _, _, _),
   th(Id3, [tt_1,R,Y,Z], _, _, [original,_],_),
   th(Id4, [tt_1,R,X,Y], _, _ , _, _),
   get_th_id(Id1).
```

The second rule is the formal version of the following informal rule:

```
if all instances of type X have a relation of the type r to an instance of the type Y,
and all instances of type Y have a relation of the type r to an instance of the type Z,
and r is transitive
then all instances of type X have a relation of the type r to an instance of the type Z.
```

```
th(Id1, [tt_2,R,all,X,an,Z], _, _, [derived], [Id2,Id3,Id4,2559]) :-
   th(Id2, [tt_2, has_characteristic, all, R, an transitivity], _, _, _, _),
   th(Id3, [tt_2,R,all,Y,an,Z], _, _, [original,_], _),
   th(Id4, [tt_2,R,all,X,an,Y], _, _, _, _).
   get_th_id(Id1).
```

2. Rules for the inheritance of features

The derivation rule for inheritance expresses that subtypes of a type having a feature probably also have that feature. For instance, if dogs have four legs, German shepherds probably also have four legs. The degree of probability depends on whether the concept is defined or is primitive (i.e., it is the source of the concept and as such not represented in Dipe-R), and on the source of the belief expressing the leg-feature. The inheritance of features is also called specialisation.

The general principle of inheritance is as follows. For a given generic concept (i.e., a type), the instances have a given feature. It then is derived for a subtype that its instances also have the

56 Since it concerns derivation, we use the term "likely".

feature. The rationale behind inheritance is that the organisation in type hierarchies, in combination with inheritance, offers a compact way of representing the features of the concepts in the hierarchy: it is no longer necessary to remember a feature for several concepts, but only once, assuming that the type hierarchy is known. The features of a subtype then can be derived from the features of a supertype, and thus do not have to be remembered explicitly.

This mechanism seems to work well for defined concepts, in a well-defined subject area. Unfortunately, in real-world subject areas, the concepts often are primitive concepts (typically natural kinds), and many features cannot be derived straightforwardly by inheritance. Hence, the potential beliefs resulting from inheritance are sometimes wrong. This may, e.g., occur if a concept was classified as a subtype of a concept with which it shares fewer features than another concept (Brachman, 1985). Therefore, for primitive concepts the features often must be remembered explicitly. As such, the inheritance mechanism should not be trusted blindly in real-life subject areas. In Dipe-R, the inherited thoughts (and all other derived thoughts) can be recognised among other represented thoughts by means of the source information (see Section 5.6).

The inheritance of features, as a way of using the subtype relation for deriving features, is closely related to the abstraction and the generalisation of features. We briefly discuss these two mechanisms.

The abstraction of features could be served by a rule complementary to the above-mentioned rule: if a concept has a specific feature, the concept also has a supertype of that feature, i.e., a more general feature. For instance, if dogs have red tongues, they also have tongues. Such a rule can be conceived of straightforwardly. Nevertheless, it is not included in Dipe-R, since the information it provides can easily be obtained in a different way[57], and it has shown to slow down performance significantly (because it yields many potential thoughts, often useless).

The generalisation of features could be served by a rule loosely formulated as follows: if all the subtypes of a concept have a particular feature, then that concept has the feature as well. For example, if there exist two kinds of glass, red and blue, and both these kinds are transparent, then it can be argued that (all) glass is transparent. The generalisation is not included in Dipe-R, since it would require the expression that all subtypes of a concept have the same feature; Dipe-R does not allow for such an expression.

Dipe-R does support inheritance. The idea of inheritance can be expressed in the following semi-formal rule:

if a has feature b
and the concept c is subtype of the concept a
then c has feature b

Given our thought types, a feature of a concept is anything related to that concept by the second sentence type. Thus, a feature is more than what is explicitly called a feature (or property, or characteristic). Our rule for derivation works with the idea of implicit and explicit features. It works as follows:

57 For instance, in Dipe-D (see Chapter 5), by looking for "any type of dog that is related to any type of tongue" one also obtains dogs having red tongues, since one asks for *any type of* tongue.

if all instances of a R q b (in which R is not the subtype relation type)
and the concept c is a subtype of the concept a
then all instances of c R q b.

Formally, the rule is expressed as:

```
th(Id1, [tt_2,R,all,X,Q,Z], _, _, [derived], [Id2,Id3,2560]) :-
   th(Id2, [tt_1,is_subtype_of,X,Y], _, _, _, _),
   th(Id3, [tt_2,R,all,Y,Q,Z], _, _, _, _),
   get_th_id(Id1).
```

5.8 Chapter evaluation

The chapter is evaluated, in a discussion (Subsection 5.8.1), a conclusion (Subsection 5.8.2), and suggestions for future research (Subsection 5.8.3).

5.8.1 Discussion

We discuss two issues related to Dipe-R's design: expressiveness and reusability.

Expressiveness of Dipe-R

In Dipe-R, the possibility to distinguish between primitive and defined concepts – as mentioned in Section 5.3 – is included, although there are bordercases. The latter can be seen as follows. Dipe-R allows for the classification of concepts into defined and primitive, by its source information. It also allows for the provision of conditions, by means of source information (see Section 5.6): if a feature has "defined" as source, the feature is a condition of the concept. However, Dipe-R does not support the expression that a collection of conditions is sufficient for the definition of a concept. We regard the inclusion of a means for expressing sufficient conditions not a high priority for supporting query formulation. The necessary conditions, that can be expressed, are expected to yield sufficient possibility to describe concepts by their features.

Moreover, Dipe-R does not have the possibility to combine represented thoughts, for example by conjunction, or disjunction. Neither does it offer the negation operator, or the expression of higher order relations. Nevertheless, we believe that Dipe-R offers sufficient expressiveness to be useful for supporting query formulation even without these possibilities. Yet, sometimes it is useful to be able to represent collections of thoughts. For instance, it may be desired to express that a collection of thoughts constitutes the necessary and sufficient conditions for a defined concept (as mentioned above), or to include a script (in the sense of Schank (1975), i.e., as a description of a typical sequence of events). Therefore, the extension of Dipe-R with such possibilities may be considered at a later stage.

The reusability of representations

The reusability of representations is a prominent research issue; see, e.g., Gruber (1991) and Gruber (1993), and Van Hyfte *et al.* (1999). So far, the reusability of representations has been a major bottleneck in the development of expert systems (Williams and Bobrow, 1991). The creation of

reusable representations is considered a way of reducing the amount of effort to formulate adequate representations.

The source information in Dipe-R allows for reusable representations: different as well as identical thoughts can be represented and then still be distinguished, on the basis of the source information. Thus, it is no longer required that only one view is represented on a subject. Everybody's view can be represented. Moreover, if a person revises his view, this can be represented (thus supporting time dependence) by adding a new represented thought; the old thought does not need to be removed (due to its time stamp it is clear that it is an older thought). Although in our research the reusability of Dipe-R is not investigated quite thoroughly, it seems a very promising way towards reusable representations.

5.8.2 Conclusion

The chapter assesses our second research question: "*Is it possible to formulate a representation framework for domain knowledge that effectively supports initial query formulation processes?*" As a result, Dipe-R is developed. The design of Dipe-R is based on existing representations, and the way we view the concept of knowledge as expressed in Chapter 2. Our view is supported by the experiences with two example domains (see Chapter 7).

Two requirements are imposed on Dipe-R, that we regard important for effective query formulation and that are not met by the existing representations reviewed. The requirements are:

(1) a proper expression of mental states and processes, i.e., (1a) expressions with clear meanings to (almost) untrained users (essentially independent of their context, i.e., maintaining the validity in many contexts), (1b) recognition of the subjectivity and time dependency of mental states and processes, and (1c) recognition of the origin of mental states and processes;
(2) the identification of query keys by the hierarchical relations and features of the concepts they designate, i.e., (2a) an adequate relation of concepts to their sub/supertypes, and to their features, and (2b) an adequate relation of concepts to terms, words, etc (linguistic elements).

Dipe-R's design meets the requirements by allowing for the representation of thoughts; two types of represented thoughts are distinguished. They have as elements: content information and source information. The content information comprises represented concepts for quantities, relations, symbols, and for other concepts. The source information regards person, moment, and origin. Additionally Dipe-R offers a (simple) derivation procedure and derivation rules for transitivity and inheritance.

Since we have been able to design Dipe-R which meets in theory all requirements set to it (see above), we may conclude that the answer to the second research question is in the affirmative. Of course, the answer is based on theory. For an empirical impression of Dipe-R, we refer to Chapters 7 and 8. Chapter 7 reports on the creation of an example representation in Dipe-R, supporting the experiments on the query formulation tool of Chapter 6. In the experiments, the tool consults the example representation; the results are reported in Chapter 8.

5.8.3 Suggestions for future research

We suggest six directions for future research related to Dipe-R.

First, the present design of Dipe-R constitutes a solid basis for more elaborate and expressive representation languages. Such languages will include more thought types than the two offered

in Dipe-R. New thought types may include elements found in existing representation languages, such as quantities having numbers (Borst, Top, and Akkermans, 1994) and combined or negated expressions as already discussed above. The elements known from related research, such as numeric certainty factors, context, second-order statements, scripts, cases, or prototypes, may be useful for various purposes.

Second, thought types may be added for restrictions to other thoughts. In database systems, such restrictions are represented in the data dictionary (Date, 1990). For instance, a restriction could be that for each person precisely one name is allowed. Storing it in the data dictionary has the disadvantage that the restrictions are not part of the representation itself, but represented on a metalevel. In Dipe-R, all thoughts are represented on the same level, hence identifiable by their hierarchies and features.

A third useful extension could be the addition of new derivation rules. Such rules might allow for the derivation of variants to designations. The variants do not need to be explicitly predefined. For example, names occur with and without middle names, and with first names in full or abbreviated. As a second example, many combinations of words may occur with as well as without hyphens. For a user formulating a query, it is laborious to conceive all variants, while they can be easily produced by appropriate derivation rules.

A fourth extension might constitute the derivation rules using type hierarchies. For example, if it is represented that all instances of Brayton cycles comprise an instance of isobaric compression, and that Brayton cycle is a subtype of thermal cycle, then it can be derived that *some* instances of thermal cycle comprise an instance of isobaric compression. This, of course, yields many irrelevant derivations next to a few relevant ones. However, sometimes the relevant derivations are sufficiently important to make it worth the effort of inspecting many derivations[58].

A fifth extension of Dipe-R may reside in the inclusion of a relation type that relates concepts to their (conceptual) parts. We suggest that the relation type be named "is_conceptual_part_of". This relation type is present in UMLS (1994). In combination with appropriate derivation rules, it will be easier to associate. For example, the rather complex concept 'temperature sensor in active feedback controlled VCHP' may be represented, along with its features. The concept 'active feedback' is considered a conceptual part of this concept. In the present Dipe-R it is not possible to identify a concept related to 'active feedback'. Once the new relation type is added, it allows for the derivation of the aforementioned features to be related to 'active feedback' (as well as to the complex concept). Of course, the relationship is indirect. Yet, the relationships derived from it may be useful.

Our sixth and final recommendation concerns research on the nature of mental states and processes, e.g., whether thoughts can be decomposed, as we assumed for Dipe-R (see Sections 5.3 and 5.4). Moreover, it could be illuminating to investigate to what degree thoughts and their expression can be distinguished; in Dipe-R, we assume that this is possible to a high degree (see Section 5.5).

---

58 In Subsection 5.7.2, we suggested a second kind of additional derivation rule using type hierarchies: a rule for the abstraction of features. The merit of such a rule depends on the tool that consults Dipe-R. However, as already mentioned, the tool of Chapter 6 as an alternative way of finding abstracted features itself (viz. by using a sub-expression with "any type of", see Section 6.3).

Chapter 6

Dipe-D: a device for query formulation

This chapter assesses the third research question, i.e., *"Is it possible to formulate a knowledge-based technique that effectively supports initial query-formulation processes?"* For this purpose, it specifies a device (Dipe-D) that supports its users in the formulation of initial queries. Dipe-D is based on our ideas on query formulation developed in Chapter 4: a two-step process of concept identification and subsequent transformation of the concepts identified into a query. This chapter works out both steps. The first step is accomplished as an iterative loop containing: (1) specifying concepts by the user ("question"), (2) solving the specification, and (3) returning concept identifiers by Dipe-D ("answer"), and (4) explaining the answer by Dipe-D. The second step is accomplished by an automatic procedure, which finds query keys to the concepts and connects the keys by operators. Both steps consult represented knowledge expressed in Dipe-R.

The chapter is structured as follows: Section 6.1 provides our ideas on query formulation as a two-step process. Sections 6.2 through 6.5 work out the first step. Section 6.6 works out the second step. Section 6.7 evaluates the chapter.

6.1 Query formulation as a two-step process

For assessing the third research question, the chapter describes a device that supports initial query formulation by consulting a representation of domain knowledge as expressed in Dipe-R. The device is named Dipe-D. Again, "Dipe" stands for Description and Identification (of concepts) by PropertiEs, i.e., the function of the first step. "D" stands for Device (as opposed to the "R" of Representation in Dipe-R in Chapter 5).

In Chapter 3, we noticed that in research on query formulation mostly the reformulation of a given initial query has been investigated. The initial query is often handmade. For example, in some approaches an initial query is formulated manually using terms identified in the exploration of a thesaurus or a similar knowledge representation. Explorers for this approach are described in Aboud e*t al.* (1991; 1993), Roziç and Dimec (1994), Belkin, Marchetti, and Cool (1995), Wiesman (1998), and Fowler, Fowler, and Williams (1998). The latter two of these approaches provide some support in formulating a (simple) query in addition to offering query keys. Other

approaches start from a given initial query and reformulate it, either before or after matching; examples of this approach were discussed in Chapter 3.

In contrast to the existing approaches, our approach for query formulation supports the whole initial query formulation process. It is based on the lessons learned in Chapter 4. In particular, it supports the two-step approach suggested in that chapter:
– developing and expressing the information need by exploring represented knowledge, and
– transforming the expression of the information need into a query.

6.2 The first step: concept identification

We discuss research relevant for the first step in Subsection 6.2.1, and then introduce our ideas for this step in Subsection 6.2.2.

6.2.1 Related research

The goal of the first step is to develop the information need and to express it in a proper way. As will be clarified below, in Dipe-D an information need is expressed as collection(s) of concept identifiers (rather than collection(s) of terms). Researchers in related research domains have developed tools that support the user in the exploration of represented domain knowledge. Below we discuss these tools, since they can be used for the functioning of our first step: by exploration, a user may gather domain knowledge and develop the information need. Well-known domain explorers can be categorised into two main types: (alphanumeric) list presentations, and graphical (network) presentations.

Explorers using list presentations have been present in many (university and public) libraries since several years. Typically they show a thesaurus as a list of terms. Sometimes the terms are arranged according to hierarchical relations and with indications for the other relations (see, e.g., WinSpirs[59]). List presentations provide information in a useful manner (see Wiesman, 1998). However, it is generally believed that the graphical presentation of terms and their relations as a network is more promising.

This feeling can be supported by the view that a graphical presentation enables the user to grasp a number of messages at once, in particular if the messages concern related information. This has to do with the high capacity of users to process spatial information (e.g., Coll, Coll, and Thakur, 1994; Hemmje, 1994). For example, in an experiment by Coll *et al.* (1994), the trade volume of a company for a series of years was shown to a test group, in a table (i.e., a list presentation) and in a graph. Most users of the test group derived the trend of the trade volumes faster from the graph than from the table[60]. Analogously, a graphical presentation of domain information may enable one to derive information rather fast (in comparison with a list presentation), e.g., whether a concept is related to many other concepts, or whether it has many subtypes.

---

59 WinSpirs is a commercially available text retrieval system from the supplier Silverplatter. Since no literature reference is available, we refer to Silverplatter's homepage: http://www.silverplatter.com.

60 The difference was largest for users with high analytical capabilities.

The graphical network presentation of domain information for Information Retrieval was studied by among others McMath, Tamaru, and Rada (1989), Aboud *et al.* (1991; 1993), Hemmje (1994), Rosengren (1994), Zavrel (1996), Fowler, Fowler, and Williams (1998), and Wiesman (1998)[61]. The domain presentations mentioned are essentially presentations as networks. They differ in the specific way of drawing. The presentations show terms and their hierarchical relations; some show also other types of relations.

McMath *et al.* (1989) show, in a two-dimensional format, terms with their hierarchical relationships: the top term of a hierarchy is surrounded by concentric circles, representing successive subtypes. Aboud *et al.* (1991; 1993) show documents and their relations (cross-references) as boxes and lines, respectively. Rosengren (1994), and Roziç and Dimec (1994) show terms and relations, as boxes and lines. Hemmje (1994) shows a three-dimensional image of the documents and domain elements, as bars and rods.

The explorer by Wiesman (1998), i.e., the ANS browser introduced in Subsection 1.2.3, shows documents and a domain, in two dimensions, also as boxes and lines (see Figure 6.1). It uses the algorithm for drawing directed acyclic graphs (DAGS) by Eades (1984); this is a heuristic algorithm that divides nodes over the screen by a metaphoric mechanical model. The latter treats the relations among nodes as springs, with an idle-length determined by the strength (or type) of the relation.

We consider the ANS browser as an appropriate example of the state of the art in graphical network explorers, since it is well devised and has been implemented and tested. In this explorer, a user is allowed to browse a domain by selecting a single node (which represents a designation or a document), and giving a command to show related nodes. The latter nodes are drawn such that they surround the selected node. The selection of a node and the subsequent presentation of its related nodes can be repeated many times and also undone if desired. This technique enables the user to focus on a part of the domain information shown, and to expand or diminish the highlighted part.

For improving the survey-ability of the screen, the nodes are divided evenly over the screen, still regarding the requirement of showing the selected node surrounded by its directly related nodes. For the purpose of survey-ability, certain relation types and concepts can be suppressed, by the activation of a filter. The filtering of relation types has proven useful for increased survey-ability. Since the browser keeps track of all actions in a session (i.e., the commands given and the images shown), the user is enabled to browse in a more directed manner. We consider it a disadvantage that the subtype relations are shown in the same font as other relations. It would have been intuitively more appealing to show them in a different manner, since human beings regard them as different.[62]

Fowler, Fowler, and Williams (1998) describe an explorer with a graphical presentation and a visible exploration of a network. The network is a DAG, created with the help of a statistical text analysis. The network contains concepts and relations (both obtained on the basis of co-occurrences of words in text documents); the relations have no type information (as opposed

61 Earlier publications about the same explorer are Wiesman, Hasman, and Van der Meulen (1994) and Dietz, Van der Pol, and Wiesman (1997).

62 In Subsection 6.7.3, we make a suggestion for such an improvement.

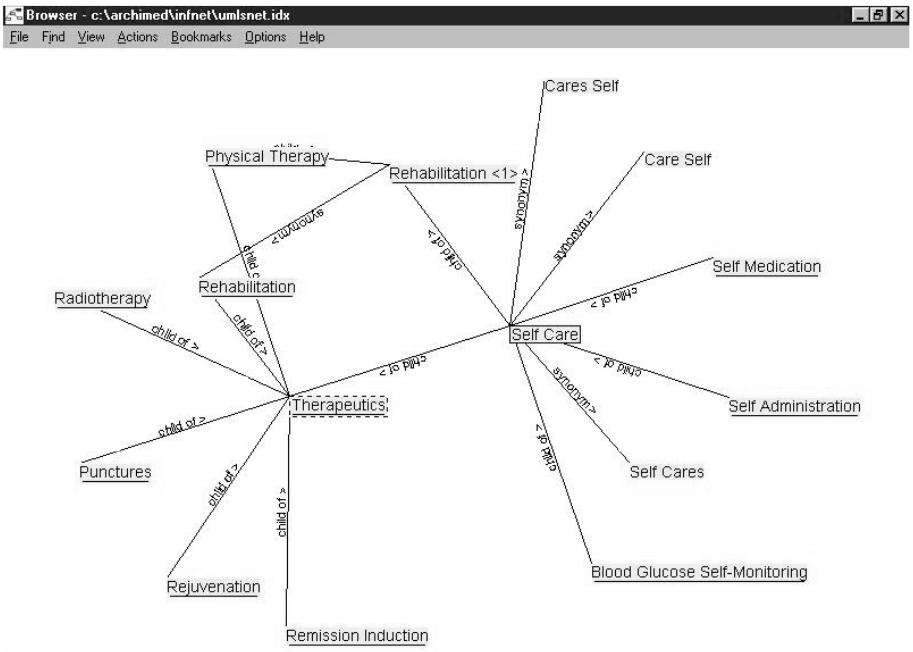

FIGURE 6.1: *The domain browser according to Wiesman (1998), showing the concepts related to therapeutics, and those related to self-care (using the UMLS[63], version 1994).*

to, e.g., the relations in UMLS). The graph presented is created using the algorithm of Eades (1984), and is in similar to Wiesman's – at least in the properties relevant to our discussion. The graph can also be explored in a similar manner. No filtering mechanism is mentioned, and no implementation is reported.

As said before, the above-mentioned list and graphical network presentations differ somewhat in their specific appearances. Yet they share the same working principle. All presentations show a network of terms connected by their relations, and offer commands for expanding and diminishing the network. The commands carefully expand the network, i.e., relation by relation, and not by large subnetworks (e.g., by a whole hierarchy). Thus, having a presentation on the screen the user is prompted to explore a knowledge representation stepwise. He then explores only one or a few relations or terms per browsing action. Even though several terms are shown simultaneously, the user starts from one of these terms, and browses from term to term, following the relations. After a while, he needs other terms than on the screen and thus has to give a command for expanding the network shown, or at least for showing other terms. He thus should be led to the terms that describe his information need.

63 UMLS (1994).

So far, the graphical network explorers were endowed with thesauri, or with UMLS (Wiesman, 1998)[64]. We believe that these explorers do not support concept identification adequately. The reason for this is that the thesaurus does not allow for the identification of terms or concepts by their features, since they lack relation-type information. Even when a more elaborate representation is used, such as UMLS, or a representation in Dipe-R, the explorers still seem to have disadvantages. We recognise the following four obstacles for their effective and efficient working:

1. the network of the representation is dense, i.e., has a high number of relations per concept,
2. the information need comprises concepts each having several features,
3. the user does not know the features of the concepts with the proper level of abstraction, and
4. the information need comprises several concepts having identical features.

The first obstacle occurs when the user has to browse many relations before he identifies the proper concept(s). Such browsing may exceed the (limited) user's mental capacity. Mental capacity is defined as the availability of means for performing a (mental) task. If the mental capacity does not suffice, the accuracy of the task performance decreases and the required time for performing the task increases (Navon, 1984). Thus, the need to browse many relations may result in disorientation and in mistakes.

The second obstacle occurs when the features must be brought into connection with the concept on the screen. It seems that this can be done by the connection of one feature at a time. Thus, the user then has to remember which concepts have the desired relation to a feature, and then has to explore the second feature (and third). This also causes a high mental load, and may lead to mistakes.

The third obstacle occurs when a longer exploration path is needed. For example, in a thermodynamic process (see Chapter 7), one part comprises the isobaric compression of gas. Thus, a relation is represented between the process (Brayton cycle) and the subprocess (isobaric compression of gas) as follows: "All instances of the Brayton cycle comprise an instance of isobaric compression of gas". The user may not know this relationship, and may start from compression, or isobaric compression. Thus he will not immediately find the relationship to the Brayton cycle, but will first explore the hierarchy of compression down to the isobaric gas compression. He cannot always know in advance that following the hierarchical relation will yield an answer. Moreover, there may be many other relations as well to follow. Therefore, he is likely to fail in identifying the appropriate relationship and concepts. Of course, he may also succeed, e.g., if both types of compression are shown in the network on the screen simultaneously and the relation to the Brayton cycle too, but often this will only occur after (repeated) change of the part of the network shown on the display. If the user does not follow many relations, he will not find the proper concepts.

The fourth obstacle occurs when several paths have to be followed. After the first concept is

64 We note that the browser by Wiesman *et al.* (1998) was not solely developed for concept identification; in its network documents are presented, next to terms and relations. As such, it offers advanced browsing capabilites, in which the user is enabled to inspect documents while browsing the domain representation and thus to obtain relevance feedback.

identified (in this process the above three obstacles may occur), it has to be remembered and the other concepts must be identified by again exploring the features, and remembering what path was taken for the first concept. This puts a high mental load on the user, and may thus lead to mistakes.

The obstacles 2 to 4 seem to occur frequently, on the basis of Chapter 4 and on the basis of our own "playing" with the browser by Wiesman (1998), using UMLS. It is still an open question whether the first obstacle often occurs. We did not hit upon this obstacle in the browser by Wiesman using UMLS, but we expect that in more elaborate representations – that represent both general and detailed features – the first obstacle occurs.

In small, sparse networks the above obstacles do not occur. Then the graphical approach seems very effective. Moreover, Wiesman (1998) notices that they may be beneficial for exploring the domain for users having very little knowledge of the domain.

Concludingly, under the given conditions the above explorers seem ineffective. We could try to improve these explorers. Instead, we provide an idea for an alternative approach for the first step that should be more effective when the afore-mentioned obstcles occur. This approach is elaborated below.

6.2.2 Ideas for the first step

Our approach to concept identification is iteratively creating a description of the concepts of the information need on the basis of their features, and then solving the description. By repeatedly providing a description and obtaining an answer and an explanation, the user should learn about the domain, and thus develop his information need. Below, we introduce the basic variant of the concept identification loop. Thereafter we describe an extended version.

Basic concept-identification loop

Our first step is based on the general idea that people learn by posing questions and getting answers. The learning process is enhanced by linking the answers to an explanation. On the basis of this idea, the first step comprises an iterative loop of questioning, answering, and explaining. The loop has four steps:

1. specification,
2. disambiguation,
3. solution, and
4. explanation.

The question is posed using an artificial language specifying the information needs. So we need a specification language (step 1). This language allows for a description of the information need, and also for (some) association. It assumes a structure of the represented thoughts of type hierarchies and features, as in Dipe-R. If the expression contains an ambiguous designation, Dipe-D asks for the proper concept of the designation (step 2). Next, the specification is matched, i.e., concepts are identified (step 3). In this process, a knowledge representation is consulted according to Dipe-R. The explanation is given on the basis of the solution process (step 4).

The specification language consists of a small collection of sub-expression types, by which a variety of expressions can be created. The specification language allows for both a broad

description and a narrow description. The underlying idea is that the user may express an immature information need by a broad specification. By using the answers, he may gather the knowledge needed to develop his information need and to create a proper expression thereof. The narrow descriptions of the specification language then may be used to express precisely an established information need, yielding only the concepts needed. In comparison with the graphical explorers discussed in Subsection 6.2.1, the specification language allows for a more powerful way of exploring. Whereas graphical explorers are based on the subsequent exploration of one or a few relations at a time, the specification language enables the simultaneous exploration of several relations. The specification language thus may simplify the exploration, and thus may make it more effective. In particular, it should even enable concept identification if the obstacles of Subsection 6.2.1 occur. Moreover, a single description may include the identification of several concepts.

The specification language of Dipe-D and its semantics (steps 1 and 3), the disambiguation (step 2), and the explanation (step 4) are worked out in the Sections 6.3 – 6.5, respectively.

Extended loop

The loop described above is the basic loop of concept identification. Each concept identifier found in the loop entirely matches the expression (of the information need). If the knowledge representation does not contain concepts that entirely match the expression, a more elaborate process is required. The latter process consists of a repetition of the basic process. It is outlined below.

The separation of the query composition process into two distinct steps requires a description of the information need that can be passed on from the first step to the second step. The description should precisely indicate the concepts about which information is needed. For describing concepts, we use collections of (identifiers of) represented concepts.

For example, if a user needs information about several types of "furniture", and knows the types "bench", "chair", and "table", he may specify his information need by the collection {bench, chair, table}. Each concept identifier in this collection represents a concept of the information need. In the second step, a query is formulated with this collection of concept identifiers. Each concept identifier in the collection fully represents a concept of the information need (hence the name "fully representing concept identifier").

If there is no concept identifier in the representation that fully represents a concept of the information need, a slightly more difficult approach is required. In such a case, the concept of the information need is described by more than one concept identifier. In the above example, such a description could be made of "furniture" or a specific kind of furniture, e.g., chair. Assume the user (loosely) describes "chair" as "a kind of object or thing, meant to sit on (by a person), and usually comprising of legs and a seat". He will then select concepts from the description; for example "object", "leg" and "seat". The information need consequently is (not unambiguously) described by the collection {object, leg, seat}[65]. Each concept identifier in the collection represents only a part of a concept of the information need (hence the name "partially representing concept identifier").

65 We note that the actual collection consists of three concept designations; only for the example we use words from natural language. In Section 4.6, we use concept identifiers.

The collection of partially representing concept identifiers is usually more complex than the one in the above example. Often, for each of the concepts in this collection, a number of alternative concepts exist, which are equally useful. This happens, e.g., if "object" and "thing" are both suitable, as alternatives. The collection then becomes {{object, thing}, leg, seat}. Thus, the collection of partially representing concept identifiers may be a collection of collections.

Based on the above, the description of the information need is either a collection of fully representing concept identifiers, or a collection of partially representing concept identifiers. It seems obvious to try and identify a collection of fully representing concepts first: {c1, c2, …, cn}. If this fails, then a collection of partially representing concept identifiers should be created: {{c11, c12, …, c1k}, {c21, c22,… c2l}, …, {cn1, cn2, … cnm}}.

The user is advised to attempt first to make a collection of concepts of which each concept entirely matches the information need. Only if such concepts are not available in the representation consulted, he should attempt to make a collection of partially representing concepts[66]. Thus, depending on the type of collection, the above iterative loop of identifying concepts is executed once or several times.

6.3 A language for the specification of information needs

In this section, the language for the specification of information needs is introduced. First, in Subsection 6.3.1, the general structure is given. In Subsection 6.3.2, the main expression types are presented, and in Subsections 6.3.3 through 6.3.7 the sub-expression types. Next, Subsection 6.3.8 discusses the negation operator. Finally, Subsection 6.3.9 provides an interface between Dipe-D and Dipe-R.

6.3.1 General structure of the language

For the general structure of the specification language, we impose two requirements:

1. the specification language must have sufficient expressive power to express the various information needs of users, in distinct degrees of precision, and
2. the mental load put by the specification language on its users must be acceptable.

Both requirements are in service of an effective search process. The first requirement was already motivated in the explanation of the basic concept identification loop (see Subsection 6.2.2). The second requirement serves to increase the efficiency and indirectly also the effectiveness.

An expression of the specification language consists of a main expression, and at least one sub-expression. The specification language offers three main expression types and seven sub-expression types. As will become clear below, these expression types satisfy the first requirement above.

---

66 If a collection of fully representing concept identifiers can be found, it may still be useful to identify partially representing concept identifiers; this may increase the recall. However, since the required effort for the latter type of collection is significantly larger than for the former type of collection, it is often not cost-effective to do so.

As described in Section 6.2, the specification language allows its users to specify the concepts needed on the basis of their features. In the domain representation consulted by Dipe-D, the features of a concept are to be represented by binary relations between concepts (and not by, e.g., attributes). In line with this, the sub-expression types introduced in this chapter are based on relations to other concepts, as, e.g., present in representations according to Dipe-R. Hence, the concepts needed can be specified by their binary relations to other concepts. The latter concepts can be given in a given type hierarchy, thus using the organisation of concepts in hierarchies. For the latter distinction, the specification language offers two variants of sub-expression types: in the first variant single concepts are designated, and in the second variant entire type hierarchies are designated (by "any_type_of designation", with "designation" being a designation of a concept). The specification language thus allows the user to specify concepts by their relations to other concepts, and by their relations to concepts in given type hierarchies. In the expressions of the specification language, the types of the relations can be specified or left unspecified. By these alternatives, Dipe-D has powerful sub-expressions. Additionally, there are sub-expression types meant for association. The major sub-expression types of Dipe-D specify the concepts of an information need as follows:

– by designation, or
– by a relation (R) to another concept (C), in which
  – R is specified by type or left unspecified, and
  – C is specified by designation or by type hierarchy.

Regarding the second requirement, i.e., the acceptable mental load on the user, we consider the specification language to be an essential part of the device's user interface. User interfaces are divided into three categories, depending on the way they communicate with the user: physical, logical, and conceptual (Siau, Chan, and Wei, 1995). Physical interfaces require that the user knows what data structures[67] are used. Logical interfaces require that the user knows the organisation of the data and the possible relations between the data elements, but the user does not have to know the data-structures that implement them. An example of a logical interface is the database query language SQL. Conceptual interfaces require the user to know what entities and relations are dealt with. They do not require that the user knows the organisation of the data.

It seems that the effort needed to express the user's information need in a conceptual interface is less than the effort needed to express the same information need in a logical interface. This thought is supported by the experiments of Siau *et al.* (1995). For reducing the effort needed to express the user's information need, and hence minimising the mental load, we have chosen the specification language to be a conceptual language.

Four other properties of the specification language are expected to contribute to a low mental load on its users: the specification language is context free, its expressions resemble natural English, the number of expression types is small, and most of the expression types have simple and intuitive semantics.

Below, the language is presented by its main expression types and sub-expression types. Each of the sub-expression types is introduced informally, after which a formal expression in

67 i.e., symbol structures, see Chapter 2.

first-order predicate-logic (FOPL) notation is given. We have chosen FOPL, since it is well-known and has clear semantics. In examples of the expression types, we nevertheless have adopted a graph notation, since this seems easier to read.

6.3.2 The main expression types

The specification language includes three main expression types. The first main expression type is the start of any specification. It is a formal version of the common form of expressing an information need "I want information on anything that…", with anything being concepts in the domain representation. The first main expression type asks for the substitution of exactly one sub-expression. The second main expression type serves to specify a collection of partially representing concepts (see Section 6.2); it starts the specification of a new concept collection. It demands for the substitution of one sub-expression and recursively substituting a main expression. The third main expression type serves for a collection of fully representing concepts as well as a collection of partially representing concepts: it adds a requirement to the specification. Likewise the second main expression type, it demands for the substitution of one sub-expression and recursively substituting a main expression. In general, an expression is created by the substitution of one or more sub-expressions.

The main expression types for an information need are, in Backus-Naur Form notation:

<information need> ::= <expr>
<expr> ::= <e1> | <e2> | <e3>
<e1> ::= **I_want_the_concepts_that** <subexpr>
<e2> ::= <expr> **and_the_concepts_that** <subexpr>
<e3> ::= <expr> **and_that** <subexpr>

The sub-expressions, <subexpr>, are defined in the Subsections 6.3.3 through 6.3.8.

The collection of concepts that meet an expression of type "e1" is the collection of concepts that meet its sub-expression "subexpr1". The collection of concepts that meet an expression of type "e2" is the union of the collections of concepts that meet its main expression "expr1" and its sub-expression "subexpr2", respectively. The collection of concepts that meet an expression of type "e3" is the intersection of the collections of concepts that meet its main expression "expr1" and its sub-expression "subexpr2", respectively.

Formally, the semantics of the expression types is as follows[68]. By notation "MP[A] = B", in which

"MP" is the matching procedure (step 4 of the basic loop in the first step of Dipe-D),

"A" is a precondition, and

"B" is a postcondition,

we denote that, after running a program implementing MP starting with precondition A, the postcondition B holds when (and if) the program terminates. Further, the variable(s) in an

68 In Subsection 6.3.8 we add the negation operator and its semantics.

expression are denoted in parentheses, like the parameters of a function (e.g., "e1(subexpr)"). Moreover, "T" denotes the contents of the domain representation consulted by the matching procedure.

MP[T,e1(subexpr)] = MP[T,subexpr]
MP[T,e2(expr,subexpr)] = {MP[T,expr], MP[T,subexpr]}
MP[T,e3(expr,subexpr)] = MP[T,expr] ∩ MP[T,subexpr]

A generally-accepted view of communication between people, offered by speech-act theory (Searle, 1969; Searle and VanderVeken, 1985), is that each utterance of a sentence in human communication has a propositional content and an illocutionary force. The propositional content is the proposition being uttered, and the illocutionary force is the intention the speaker has in uttering the proposition. The illocutionary force indicates, e.g., that the utterance is a thought, a question or a command. The specification language allows for one illocution type only: the question. It is expressed in the first main expression type. In the second main expression type, it is present through the above-mentioned recursion.

As said before, the specification language includes seven sub-expression types. In the Sections 6.3.3 through 6.3.7 we introduce the sub-expression types, divided into five major kinds. The first four of these kinds are ordered according to descending degree of specificity. The fifth kind cannot be positioned unambiguously in this ordering.

The first, and most specific kind of sub-expression type designates the concept(s) needed. The second kind of sub-expression relates a given type to a given concept. The third kind arbitrarily relates a given concept. Finally, the fourth kind is that of a given relation type.

In some sub-expression types, instead of "designation", the more general "any_type_of designation" is used, allowing for the concept designated by "designation" and its subtypes. For simplicity, we introduce both variants along with each other; when looking very strictly, they have to be positioned elsewhere among the four kinds. Hence, we regard the variants with "any_type_of designation" as arranged in a dimension different from the four kinds. The variants are distinguished by a prime.

6.3.3 Sub-expression types for designation

Sub-expression type 1

The first sub-expression type is straightforward: it allows for specifying concepts by their designation. This sub-expression type is to be used if the user needs a single concept, of which he knows a designation.

<s1> ::= **have_des** <designation>

The collection of concepts meeting this sub-expression (with a given domain representation) is defined as follows:

MP[T, s1(designation)] = [T, {concept | T ⊨ have_des(concept, designation)}]

In these expressions, "concept" denotes a concept identifier, "T" denotes the theory as established by the domain representation, and "designation" is a designation (given by the user).

The sub-expression type 1 is exemplified in the following expression:

I_want_the_concepts_that have_des bank

To this expression, the device returns each concept in the domain representation with the designation "bank". If a user only makes expressions with this type of sub-expression, the device at first sight seems not to offer any benefit. However, upon closer inspection we learn that a single designation may indicate more than one concept, i.e., it may have several meanings. In such cases, using the device reveals the ambiguity; the user then may take precautions for omitting the unneeded concepts. The problem of ambiguity occurs in several sub-expressions; it is discussed in Section 6.4.

Sub-expression type 1'

In some cases a user may name a concept that stands for a single type, but actually wants the concepts that stand for its subtypes as well. Expression type 1' allows for this explicitly:

<s1'> ::= **are_any_type_of** <designation>

This expression type yields a hierarchy of concept types with the designated concept as top type. The collection of concepts meeting s1' is defined as follows:

MP[T, s1'(designation)] = [T, {concept | T ⊨ ∃concept1: have_des(concept, designation) ∨ (relation(subtype, concept, concept1) ∧ have_des(concept1, designation))}]

In these expressions, "designation" denotes a designation, and "concept" and "concept1" denote concept identifiers. "Subtype" is a constant, indicating the type name of a relation. In the case that "designation" is a homonymous name, several hierarchies may meet the specification. As said before Dipe-D may signal this, see Section 6.4.

As discussed in Chapter 5, Dipe-R defines the subtype relation as a transitive one. Through the transitivity, Dipe-D finds subtypes of the type designated by the designation "designation" as well as subtypes of subtypes etc.

The sub-expression type 1' is exemplified in an expression in which it is applied twice. The expression is

I_want_the_concepts_that are_any_type_of liver disease **and_that are_any_type_of virus disease**

This expression yields all the concepts that are in the type hierarchy of liver disease and in that of virus disease. The example domain representation is shown as a graph in Figure 6.2.

The represented thoughts of Figure 6.2 are of the second type (tt_2) of Chapter 5, and contain relations of the types "is_a_subtype_of" and "has_designation". In the figure these can be distinguished by what they connect: the subtype relationtype connects two (arbitrary) concepts, and the designation relationtype connects an arbitrary concept with a designation

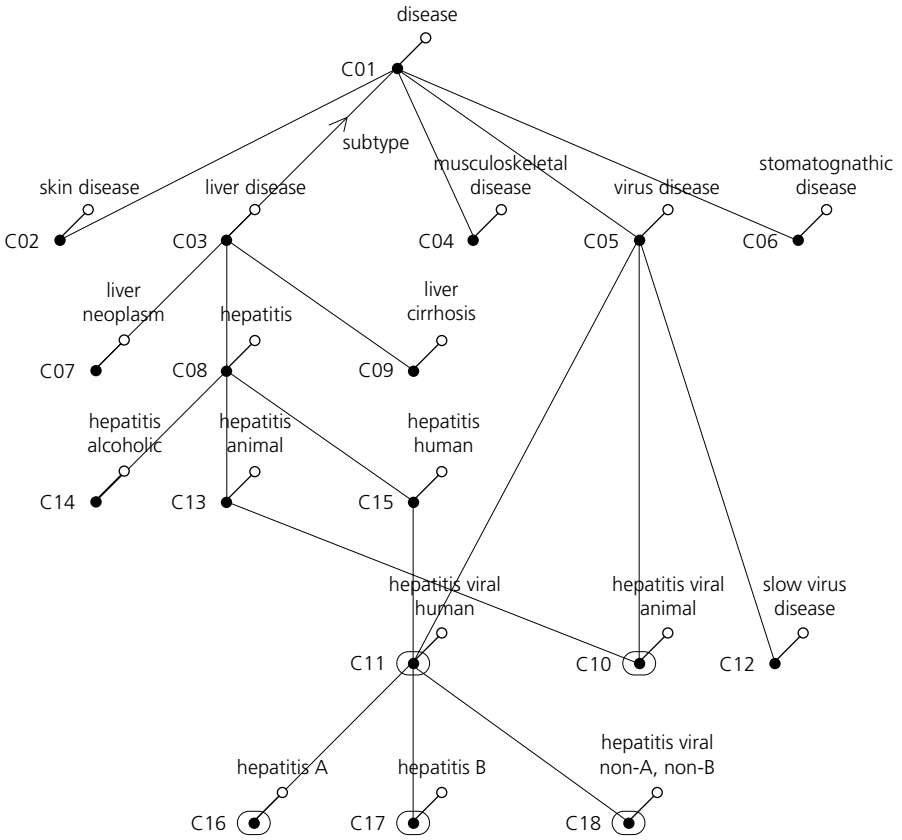

FIGURE 6.2: *Graph of domain representation, showing the lattice structure of the domain.*

concept. Concepts are represented by filled circles, and designations by open circles. The concepts are accompanied by their identifiers; a capital "C" followed by a number. The designation circles are accompanied by the concept designations. The concepts in a box meet the above example expression (c10, c11, c16, c17, and c18). Such concepts exist only if concepts exist that are part of at least two hierarchies, i.e., if the domain has an acyclic directed graph structure; this is the case in the example.

6.3.4 Sub-expression types with a given relation type and concept

If a user does not know a designation of a concept needed, a description may be made with concepts directly related to the concept needed. With the sub-expressions in this subsection, at least one such a related concept is specified, as well as the type of relation. In the second sub-expression type, the relation type is designated, in the third sub-expression type it is specified by designating another concept that has the same type of relation to the specified concept as the concept(s) needed.

Sub-expression type 2

The second sub-expression type requires a relation type name, and a concept designation.

<s2> ::= **have_a_relation_of_type** <relationtypename> to <designation>

The collection of concepts meeting s2 is defined as follows:

MP[T, s2(relationtypename, designation)] = [T, {concept | T ⊨∃ concept1: have_des(concept1, designation) ∧ (relation(relationtypename, concept, concept1) ∨ relation(relationtypename, concept1, concept))}]

Example: If a user wants concepts that somehow treat hepatitis, including those that are not a type in the treatment hierarchy, he may specify:

I_want_the_concepts_that have_a_relation_of_type treats **to** hepatitis

Two variants allow for the inclusion of the direction of the relation type:

<s2a> ::= **are_the_source_of_a_relation_of_type** <relationtypename> **to** <designation>
<s2b> ::= **are_the_destination_of_a_relation_of_type** <relationtypename> **to** <designation>

The collection of concepts meeting s2a and s2b is defined as follows:

MP[T, s2a(relationtypename, designation)] = [T, {concept | T ⊨∃ concept1: have_des(concept1, designation) ∧ relation(relationtypename, concept, concept1)}]
MP[T, s2b(relationtypename, designation)] = [T, {concept | T ⊨∃ concept1: have_des(concept1, designation) ∧ relation(relationtypename, concept1, concept)}]

Sub-expression type 2'

In some cases the user does not mean "t2" literally; any subtype will do. Sub-expression type 2' allows for this.

<s2′> ::= **have_a_relation_of_type** <relationtypename> **to_any_type_of** <designation>

Example: As in the previous example, the user wants concepts that somehow treat hepatitis, including those that are not a type in the treatment hierarchy. In this example, not only treatments of hepatitis are wanted, but also treatments of subtypes of hepatitis. He then may specify:

I_want_the_concepts_that have_a_relation_of_type treats to **any_type_of** hepatitis

In analogy to sub-expression type 2, sub-expression type 2' has two variants in which the direction of the relation is specified, in addition to the relation type:

<s2′a> ::= **are_the_source_of_a_relation_of_type** <relationtypename> **to_any_type_of** <designation>
<s2′b> ::= **are_the_destination_of_a_relation_of_type** <relationtypename> **to_any_type_of** <designation>

Sub-expression type 3

The third sub-expression type is meant for cases in which the user does not know the name of the relation type, and may or may not know the relation type; yet, he knows concepts with the same relation type. He knows that t2 is related to t1, and that t2 plays a certain role in this relationship. He is looking for another concept that fulfils the same role. He then may ask for anything that has the same relation type to t1 as t2 has. As an example, he may know one type of treatment of hepatitis-human (viz. drug-therapy), and look for other treatments of the same disease; this may be expressed as "anything that has the same type of relation to hepatitis-human as drug-therapy has".

<s3> ::= **have_the_same_type_of_relation_to** <designation1> **as** <designation2> **has**

The collection of concepts meeting s3 is defined as follows:

MP[T, s3(designation1, designation2)] = [T, {concept | T ⊨∃ concept1, ∃ concept2, ∃ relationtypename: have_des(concept1, designation1) ∧ have_des(concept2, designation2) ∧ relation(relationtypename, concept2, concept1) ∧ relation(relationtypename, concept, concept1)}]

Example: Assume that a user who knows that drug therapy has a relation to hepatitis, but who does not exactly know what type of relation, wants other concepts with the same type of relation. He may specify:

I_want_the_concepts_that have_the_same_type_of_relation_to hepatitis **as** drug_therapy **has**

The fictitious example domain is shown as a graph (Figure 6.3). A box marks the two concepts that satisfy the expression.

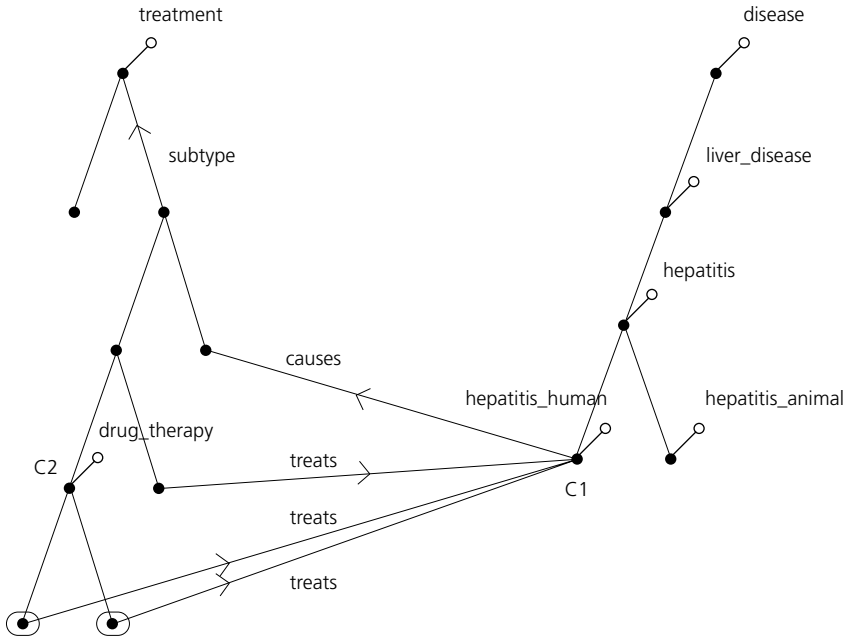

FIGURE 6.3: *Graph representation of example domain.*

Sub-expression type 3'
A replacing "designation1" in sub-expression type 3 by "any_type_of <designation1>", sub-expression type 3' is obtained. In the latter sub-expression type, "designation2" is related to "designation1", whereas the concepts needed can be related to subtypes of the concept designated by "designation1".

<s3'> ::= **have_the_same_type_of_relation_to_a_type_of** <designation1> **as** <designation2> **has**

The collection of concepts meeting s3' is defined as follows:

MP[T, s3'(designation1, designation2)] = [T, {concept | T ⊨∃ concept1, ∃ concept2,
∃ relationtypename: have_des(concept1, designation1) ∧ have_des(concept2, designation2) ∧
relation(subtype, concept3, concept1) ∧ relation(relationtypename, concept2, concept3) ∧
relation(subtype, concept4, concept1) ∧ relation(relationtypename, concept, concept4)}]

6.3.5 Sub-expression types with an arbitrary relation to a given concept

Sub-expression type 4
A less strict specification than that of Subsection 6.3.4 is obtained by leaving the relation type unspecified, and thus allowing for an arbitrary relation type to a given concept. The two sub-

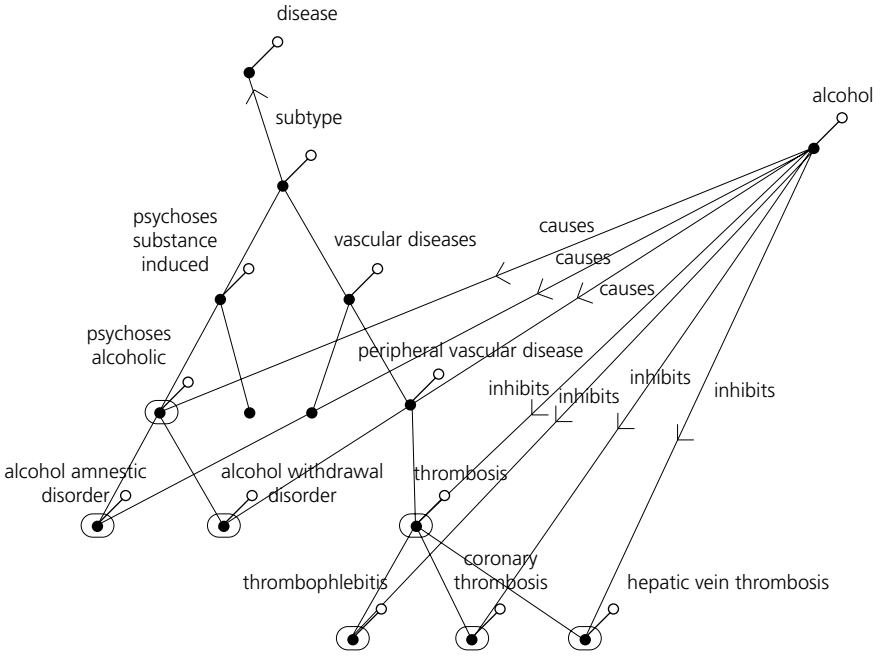

FIGURE 6.4: *Graph representation of the example to sub-expression type 4.*

expression types in this subsection allow for such specification. Sub-expression type 4 resembles "anything that has to do with t". It yields the concepts directly related to the concepts of a given designation; and also the concepts directly or indirectly related by a transitive relation type (see also Chapter 5).

<s4> ::= **are_related_to** <designation>

The collection of concepts meeting s4 is defined as follows:

MP[T, s4(designation)] = [T, {concept | T ⊨∃ concept1, ∃ relationtypename: have_des(concept1, designation) ∧ (relation(relationtypename, concept, concept1) ∨ relation(relationtypename, concept1, concept))}]

Example: When looking for disease types in which alcohol plays a role, the user may specify:

I_want_the_concepts_that are_any_type_of disease **and_that are_related_to** alcohol

Figure 6.4 shows a graph of a domain representation for the example. As in the example of sub-expression 2, the concepts that meet the specification are drawn in boxes.

Sub-expression type 4'

The above example expression yields concepts related to alcohol, but no concepts related to specific types of alcohol. If these are to be included, the name "alcohol" in the expression should be replaced by "any_type_of alcohol". This yields sub-expression type 4'.

<s4'> ::= **are_related_to_any_type_of** <designation>

The collection of concepts meeting s4' is defined as follows:

MP[T, s4'(designation)] = [T, {concept | T ⊨∃ concept1, ∃ concept2, ∃ relationtypename: (have_des(concept1, designation) ∧ (relation(relationtypename, concept, concept1) ∨ relation(relationtypename, concept1, concept))) ∨ (have_des(concept1, designation) ∧ relation(subtype, concept2, concept1) ∧ (relation(relationtypename, concept, concept2) ∨ relation(relationtypename, concept2, concept))}]

Sub-expression type 5

The user may not know the designations of the term the concepts needed are related to. If he does know another term, that is related to the term to which the concepts needed must be related, he can use sub-expression type 5:

<s5> ::= **are_related_to_the_same_type_of** <designation1> **as** <designation2>

The collection of concepts meeting s5 is defined as follows:

MP[T, s5] = [T, {concept | T ⊨∃ concept1, ∃ concept2, ∃ concept3, ∃ relationtypename1, ∃ relationtypename2: have_des(concept1, designation1) ∧ relation(subtype, concept3, concept1) ∧ have_des(concept2, designation2) ∧ relation(relationtypename1, concept2, concept3) ∧ relation(relationtypename2, concept, concept3)]

Example: If a user wants disease types that have the same treatment as hepatitis-human, he may specify:

I_want_the_concepts_that are_any_type_of disease **and_that**
are_related_to_the_same_type_of treatment **as** hepatitis-human

The example domain is shown in a graph (Figure 6.5). The three concepts that satisfy the expression are drawn in boxes.

Since relations of different types exist between the treatment and disease hierarchies, the collection of concepts meeting the specification may contain unneeded concepts, as is the case with the treatment type related by a relation of the type 'causes'. For such cases, another sub-expression type may be used, specifying the relation type as well.

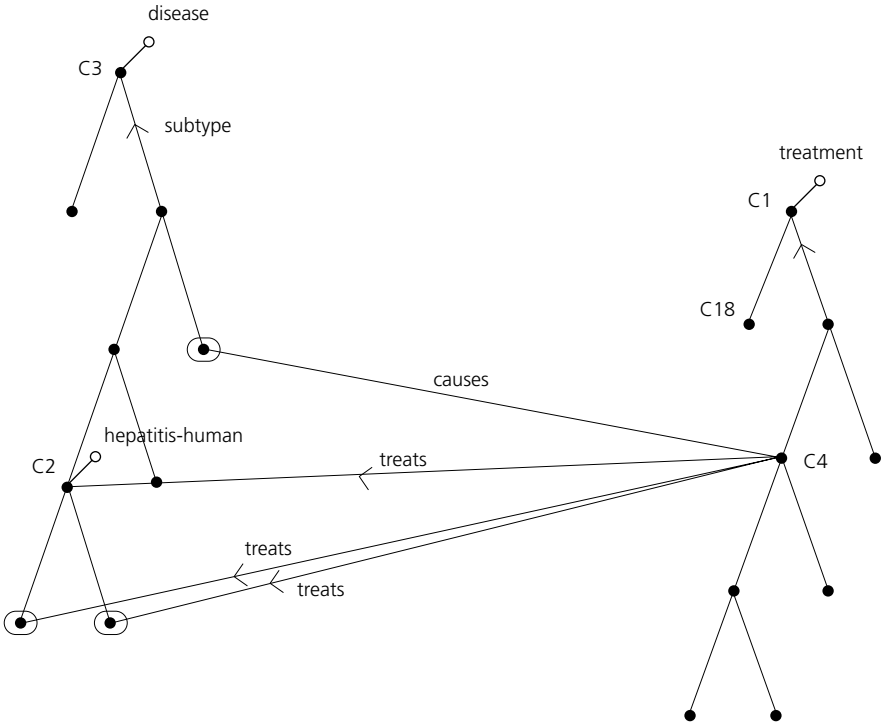

FIGURE 6.5: *Graph representation of example domain.*

6.3.6 Sub-expression type with a given relation type

Sub-expression type 6

If a user knows no related terms to the concepts needed, and does know a relation type and its name, he may use sub-expression type 6:

<s6> ::= **have_a_relation_of_type** <relationtypename>

This sub-expression type yields all concepts having at least one relation of the specified type. The relation may be a relation to any concept, and it may have the needed concepts on either the first or second position, i.e., the direction of the relation does not matter.

The collection of concepts meeting s6 is defined as follows:

MP[T, s6(relationtypename)] = [T, {concept | T ⊨∃concept1: (relation(relationtypename, concept, concept1) ∨ relation(relationtypename, concept1, concept))}]

Two alternatives are offered, that enable the user to specify the direction of the relation:

<s6a> ::= **are_the_source_of_a_relation_of_type** <relationtype>
<s6b> ::= **are_the_destination_of_a_relation_of_type** <relationtype>

The collections of concepts meeting s6a and s6b are defined as follows:

MP[T, s6a(relationtypename)] = [T, {concept | T ⊨∃ concept1: relation(relationtypename, concept, concept1)}]
MP[T, s6b(relationtypename)] = [T, {concept | T ⊨∃ concept1: relation(relationtypename, concept1, concept)}]

Example: If a user wants all treatment types, he may specify

I_want_the_concepts_that are_source_of_relation_of_type treats

Using sub-expression type 6 separately, will in general yield too many concepts. We expect this, in many domains the number of relation types is much smaller than the number of relations. Each relation type then is likely to occur many times in the domain. The use of sub-expression type 6 must be sought in the combination with sub-expressions of other types.

6.3.7 Sub-expression type with prepositions

In some cases, the user knows which type hierarchy contains the concepts he needs, and a concept (typically in a different type hierarchy) related to the concept he needs. In natural language, he then expresses his information need as "t1 preposition t2", with t1 designating the top type in a hierarchy, and t2 designating the known related concept. For example, "the treatment(s) of cholecystitis", and "the tool(s) of cholecystitis-surgery".

The prepositions express that a relation exists between the hierarchy and the known concept; the preposition limits the allowed types of relations. For instance, the preposition "of" in "treatment of human beings" allows for other relation types than the preposition "by" in "treatment by human beings". Yet, a preposition in some cases allows for several relation types.

Considering the relation type as unspecified, "treatment of cholecystitis" can be expressed by using the sub-expression types 1 and 4:

I_want_the_concepts_that are_any_type_of <designation1> **and_that are_related_to** <designation2>

We believe that users frequently express their information needs in the aforementioned way with a preposition. Sub-expression type 7 is in line with this:

<s7> ::= **are_any_type_of** <designation1> **{preposition}** <designation2>

For example, if a user needs concepts about "knives for surgery", he may express this as:

I_want_the_concepts_that are_any_type_of knife **{preposition}** surgery

Sub-expression type 7 is an alternative notation for the expression type **are_any_type_of** <designation1> **and_that are_related_to** <designation2>.

Replacing "designation2" by "any_type_of designation2" yields sub-expression type 7':

<s7'> ::= **are_any_type_of** <designation1> **{preposition} any_type_of** <designation2>

Sub-expression 7' is an alternative notation for the expression type **are_any_type_of** <designation1> **and_that are_related_to any_type_of** <designation2>.

6.3.8 The negation operator

A user may want to exclude certain concepts that satisfy a given expression. For example, he may want information on any type of hepatitis, except hepatitis-B. For such purposes, the specification language supports a negation operator. It is to be used in combination with a sub-expression. In the expression, it precedes the sub-expression.

In Subsection 6.3.2, we defined the collection of concepts that meet an expression of the specification language. This definition holds only for expressions that do not contain a negation operator. If an expression does contain a negation operator, the collection of concepts that meet the expression is defined as follows.

The concepts meeting an expression of type e1 with a negation operator are: the set-operational difference of all concepts in the domain representation, and the concepts meeting subexpr1 (of course, an expression of this kind will in practice hardly ever be created). The concepts meeting an expression of type e2 with a negation operator are: the set-operational difference of the concepts meeting expr1 and the concepts meeting subexpr2.

The collection of concepts that meet an expression of type "e1" is the collection of concepts that meet its sub-expression "subexpr1". The collection of concepts that meet an expression of type "e2" is the intersection of the collections of concepts that meet its main expression "expr1" and its sub-expression "subexpr2", respectively.

In combination with at least one sub-expression that is not negated, it may be useful for abolishing a number of concepts that were included through the other sub-expressions. The aforementioned example is expressed as

I_want_the_concepts_that are_any_type_of hepatitis **and_that_are not any_type_of** hepatitis-B

The negation operator is interpreted as "not known" that… This meets the interpretation of the absence of a represented thought in Dipe-R. We extend the definition of each of the three main expression types with an optional negation:

<e1> ::= **I_want_the_concepts_that** <full_subexpr>
<e2> ::= <expr> **and_the_concepts_that** <full_subexpr>

<e3> ::= <expr> **and_that** <full_subexpr>
<full_subexpr> ::= <subexpr> | **not** <subexpr>

Which concepts match an expression with a negation is defined as follows:

MP[T, **not** subexpr] = {concept | T ⊨ concept ∈ T} - MP[T, subexpr]

In words: all concepts in the representation, except those matching the sub-expression. With this definition, the definition of the concepts matching the main expressions are unchanged:

MP[T, e1(full_subexpr)] = MP[T, full_subexpr]
MP[T, e2(expr, full_subexpr)] = {MP[T, expr], MP[T, full_subexpr]}
MP[T, e3(expr, full_subexpr)] = MP[T, expr] ∩ MP[T, full_subexpr]

The expressions e1' and e3' should be omitted, as their implementations will be inefficient due to the need for retrieving all the concepts in the representation (T). The expression e2' is in particular meant for negations: it can be implemented such that it subtracts elements from a collection already found, instead of retrieving all the concepts. The solution is defined as "all concepts matching *expr*, except concepts that match *subexpr*".

MP[T, e3'(expr, subexpr)] = MP[T, expr] - MP[T, subexpr]

In Boolean queries for full-text retrieval, a negation operator is sometimes used to exclude unintended meanings of ambiguous designations; see also Section 6.6. Such use is not necessary in the specification language, as this language offers a different way of dealing with ambiguous designations in an expression; see Section 6.4.

6.3.9 Interfacing to Dipe-R

For consulting a knowledge representation according to Dipe-R, Dipe-D's specification language must be connected to the thought types of Dipe-R. Dipe-D uses represented thoughts having binary relations. Thoughts of this type can be derived from Dipe-R's two main represented-thought types, by the two rules listed below. Their notation is according to the convention of Section 5.6.

```
th(Id1, [rel,R,X,Z], _, _, [derived],[Id2,2549]) :-
    th(Id2, [sentt_1,R,X,Z], _, _, _, _),
    get_th_id(Id1).

th(Id1, [rel,R,X,Z], _, _,[derived],[Id2,2552]) :-
    th(Id2, [sentt_2,R,Q1,X,Q2,Z], _, _, _, _),
    get_th_id(Id1).
```

6.4 Dealing with semantic ambiguity in specifications

Concepts and their designations are not related on a one-to-one basis. Designations are made of words, and the same word may designate different concepts. For example, the (Dutch) word "bank" may be synonymous with "financial institution" as well as "piece of furniture". This phenomenon is called lexical ambiguity; it is generally divided into two types: syntactic and semantic (Small, Cottrell, and Tannenhaus, 1988). A word that may be in different syntactic categories is called syntactically ambiguous. For example, "run" occurs as a noun and as a verb. A word that may designate different concepts is called semantically ambiguous. Homonymy and polysemy are forms of semantic ambiguity. If the concepts are not (directly) related, we speak of homonyms, if they are related, we speak of polysemy. In practice, homonyms and polysemy are difficult to distinguish.

In searching full-text-indexed documents, lexical ambiguity is known as a source of problems (Krovetz and Croft, 1992). Both semantic ambiguity and syntactic ambiguity cause documents to be retrieved that are not relevant. Moreover, their ambiguity does not prevent them from sometimes failing to retrieve relevant documents.

Lexical ambiguity also plays a role in the expressions of the specification language. The specification language contains words, which are not always unique concept identifiers. Hence, an expression may be lexically ambiguous. Since all designations used in the specification language and in the domain representation are nouns or noun phrases, syntactical ambiguity cannot occur. Semantic ambiguity is likely to occur, however. Several designations in the domain representation will have more than one concept, even if the representation is about a specialised subject area (Krovetz and Croft, 1992).

Dipe-D contains a simple procedure that deals with semantic ambiguity of expressions. The procedure checks for each given expression whether the designations in it are ambiguous, through looking up each designation in the domain representation. If a designation has more than one concept, Dipe-D asks for a selection from these concepts.

Such selection cannot be done straightforwardly by stating the concepts intended, since concepts are identified by numbers, and hardly any user will be able to recognise a concept by a unique identification number alone. In general, the concepts to a term can be recognised by their characteristics. In terms of domain representations according to Dipe-R (see Chapter 5), characteristics are defined as relations to other concepts. Alternative designations, e.g. descriptions, may also reveal the characteristics of concepts. For enabling the user to recognise the different concepts of a designation, Dipe-D presents the (designations of) related concepts, and, if present in the domain representation, it presents the alternative designations.

Once the user has selected one of the concepts of the ambiguous designation in the specification, the concepts to the specification are found in the common way; Dipe-D uses only the selected concept. In the subsequent formulation of a Boolean query, the lexical ambiguity may return; this is elaborated in Section 6.6.

If Dipe-D selects the concepts automatically, it relieves the user of this task. In some cases, automatic disambiguation seems possible, with the context of the ambiguous designation in a given expression. In text processing, such disambiguation is done with varying degrees of success. Hayes (1977) and Hirst (1988) tried automatic disambiguation with semantic networks and constraint spreading activation; a similar approach may be tried with our domain representation,

resembling a semantic network. However, we regard automatic disambiguation as a sideline from our main goal, viz. structuring the query-formulation process. Moreover, we expect that manual disambiguation can easily be done by the user in the above manner. For these reasons, we do not elaborate on automatic disambiguation.

6.5 Explaining the solution of an expression

In the first part of a query formulation process, the concepts about which the user needs documents are identified. This requires an adequate expression of the user's information need. As described in the Section 6.2, an adequate specification is realised by a repetitive process of (a) creating a specification and then (b) identifying the concepts in the domain representation meeting the specification. This process is iterative: each successive specification is meant to be a more appropriate expression of the user's information need than the earlier specifications in the process.

Conditions for obtaining improved specifications are that the user (1) gains an understanding of his knowledge lack (see Chapter 4), and (2) adequately expresses his information need (obvious).

Both conditions can be met by providing the user with domain knowledge. For the first condition it is important to tell the user what concepts are in the domain, and how they are related. This helps the user in deciding on which concepts he wants documents. For the second condition it is important to learn the user what designations are used for the concepts in the domain representation, and what an expression of the specification language means in the domain (e.g., if he asks for all subtypes of a certain concept, he learns in the domain which concepts will be included in the solution).

Dipe-D offers domain knowledge by explaining the solution to each specification. For each concept of the solution, the thought in the domain representation that matches the specification and contains the concept is presented. Moreover, that thought's source information is provided (see Section 6.7). As will be clear from the Section 6.7, an original thought has as source information the name of the person who stated it and the moment of thought. A derived thought has as source information a moment of derivation and the derivation path. The path consists of the thoughts (rules and factual) used in the derivation process; hence, the path can be elaborate. Moreover, the path may start in very specific thoughts and end in general thoughts matching the specification. By learning such a path, the user refines his domain knowledge, and hence is enabled to refine iteratively his specifications.

The explanation as described includes the kind of explanation provided by a number of shells for expert systems (see, e.g., Buchanan and Shortliffe, 1984). In such systems, the "conclusion" derived by the system is shown (a "fact"), and the path along it was derived, consisting of facts and derivation rules. However, Dipe-D offers an extended explanation, by also providing other source information (the person who had a thought, and the moment of thought) and as such is meant to support in deciding whether to accept the solution to a specification or to deny it.

In addition to providing the source information to a solution, information can be provided for clarifying a concept in the solution. Such information is obtained by presenting on a command all represented thoughts of the domain representation containing the concept; the concept is then learned by its relations to other concepts (this is possible in the Quick Browser, see Section 7.1).

6.6 The second step: Transformation

We start this section with an overview of relevant related research (Subsection 6.6.1), and then we work out the transformation. The latter is performed into two parts: (1) fully representing concepts are described in Subsection 6.6.2 and partially representing concepts in Subsection 6.6.3.

6.6.1 Related research

The second step serves to transform the expression of the information need as one or more collections of concept identifiers into a query. Previously, we mentioned already some approaches of making a query from a collection of terms, such as Roziç and Dimec (1994) and Wiesman (1998). They provide simple queries only. More advanced techniques for query formulation start at a given query and reformulate it. The most elaborate approach seems that by Järvelin *et al.* (1996), of which experiments were reported in Kekäläinen (1998). In this approach (see also Subsection 3.4.2), a query is first decomposed into a collection of collections of concept identifiers having identical structure as that created in our first step. Subsequently, a new query is created having the designations of the concepts, including synonymous designations, and terms for related concepts, etc. Hence, this decomposition and subsequent reformulation is in effect a query expansion technique. It is parameterised, allowing for operators for different types of (full-text) matching processes, such as Boolean and probabilistic ones. As discussed in Section 3.3, this technique has proven effective in probabilistic matching.

In this chapter, we work out an approach similar to the reformulation step described above. Our approach is identical in that it finds designations for concepts, and increases the recall by including synonymous designations. Our approach differs in that it also includes quasi-synonyms, and spelling variants in the query for further increasing the recall. Moreover, our approach includes a technique primarily aiming at the enhancement of the precision (though possibly deteriorating the recall slightly). Our approach does neither add various related concepts, since it is assumed that all relevant concepts of the information need have already been identified in the concept identification step, nor is our approach parameterised; it creates Boolean queries. However, it can in essence be parameterised, since this constitutes mainly the substitution of operators[69].

Our approach is as follows. Once the identifiers of concepts needed are identified, the device transforms the collection of concept identifiers into a Boolean query. The transformation process has two variants: one with a collection of fully representing concepts (Subsection 6.6.2) and one with a collection of partially representing concepts (Subsection 6.6.3). The former variant is applied if the concepts needed are present in the domain representation; the latter if they are not present, then describing concepts are used (each of which only partially represents the user's information need).

The formulation of a query with a collection of partially matching concepts recursively calls the transformation process for concepts present in the domain representation. In each recursive call, a sub-query is made for one of the describing concepts. In this way, both variants have the same basic procedure.

---

69 When our second step was first reported (see Van der Pol, 1996), the approach by Järvelin *et al.* (1996) had not been published.

6.6.2 Transformation using fully representing concepts

The transformation process consists of the creation of a sub-query to each of the concepts needed, and the combination of the sub-queries into a Boolean query.

Basic technique

In conceptually indexed documents, no lexical variants or synonyms of the designation exist in the vocabulary; for each concept, there is precisely one designation. Moreover, a well-designed vocabulary contains no ambiguous designations. These issues were discussed in Section 3.2.

From the above, we conclude that the creation of a sub-query for a concept can be done by literally using the designation of the concept as sub-query. This is also our basic technique for full-text indexed documents. It is expressed in formula (1).

$$BQ\_concept(c) := \text{“}designation\text{”} \tag{1}$$

The information need comprises a collection of concepts (C). Each of the documents to be matched has to contain at least one of the concepts needed. This is expressed in formula (2):

$$BQ\_concept\_group := \bigcup_{c \in C} BQ\_concept(c) \tag{2}$$

The union sign symbolises the collection of disjunctions (“OR”-operators) between the sub-queries of (1).

Two recall enhancement techniques

In full-text indexed documents, the language used by the authors of the documents transpires in the index terms (see Section 3.2). This implies that, among other things, for the same concept different index terms may occur (Blair and Maron, 1985). Designations consist of words, that may have alternatives. By including alternatives of the designations and the words, more alternative designations are included and as such a higher recall is likely to be obtained. For this purpose we propose the following two techniques, called recall enhancement techniques.

The first technique is a weak recall enhancement technique. Its steps are:
– finding alternative designations of the concept (synonyms, and lexical variants),
– creating a sub-query with the individual words (and word variants, and word stems, excluding stopwords[70]) of each designation, using the formula’s (4) and (5), and
– combining the sub-queries in a query, using formula (3) and then (2).

The second technique is a strong recall enhancement technique. It is almost identical to the first technique, but differs in that it also adds quasi-synonyms next to synonyms, and lexical variants.

In the recall enhancement techniques, the variants are not used literally; instead they are added by individual words, and proximity operators connect these. For its realisation, we assume a Boolean query language that supports a proximity operator, next to the common conjunction,

70 See Subsection 3.2.1.

disjunction and negation operators, and the usual separators. Moreover, we assume the truncation operator to be valid. These operators and separators are offered by most Boolean query languages. For instance, they are offered in the STAIRS, ORBIT and MEDLARS systems, which were developed in the 1970s (Salton and McGill, 1983). More recent systems, e.g., INQUERY (Callan, Croft, and Harding, 1992), TRIP (VIP, 1994) and TOPIC (Chong, 1989) include new features, but have the above operators and separators as their basis. We denote the (binary) proximity operator as "NEAR" and the (unary) truncation operator as "*".

The designation variants are included in the query by formula (3):

$$BQ\_concept := \bigcup_{i \in I} designation(i) \tag{3}$$

Disjunction operators connect the variants to words, since it suffices if one of the variants is present. Hence, the Boolean query corresponding to a word becomes:

$$BQ\_word := word \vee wordstem^{\star} \bigcup_{i \in I} variant(i) \tag{4}$$

In relevant documents, the words of a designation are likely to appear in each others neighbourhoods. Although a few additional words may separate them, it is unlikely that they are remote and yet belong to the same designation. If they are remote, it is likely that they belong to different designations. In the query, we express that they must appear in each other's neighbourhoods by using a proximity operator. Formula (5) expresses the query belonging to a designation. The subscript indicates that the proximity operator is used for the collection of sub-queries to its right.

$$BQ\_designation(d) := \underset{i \in I}{NEAR}\ BQ\_word(i) \tag{5}$$

As already mentioned, the sub-queries to designations are combined into a query according to formula (3) and then (2).

Precision enhancement technique

Next to a recall enhancement procedure, we propose a precision enhancement technique. The latter may be used in combination with either the basic procedure or the recall enhanced procedure. Its main function is to disambiguate the designations in a query; this decreases the number of irrelevant documents matched, and thus increases the precision.

The designations used for obtaining a query may be ambiguous (Blair and Maron, 1985). This results in the matching of documents, dealing with an unintended concept of a designation, hence irrelevant documents. We describe how designations can be disambiguated.

In the specification of an information need, ambiguity was handled by selecting the intended concepts by their identifiers, as present in the domain representation (see Section 6.3). In full-text retrieval, no such identifiers are available. Therefore, the exclusion of irrelevant documents must come from different approaches.

In the study of Chapter 4, we observed an approach to the disambiguation of designations in queries in which a query is extended with a word that is related to the unintended concept. The word is preceded by the negation operator, so that documents containing the word are not

matched. For example, if the Dutch word "bank" is meant in the sense of financial institution, the query may become "bank AND NOT meubel"[71]. This approach is based on the assumption that a document dealing with the intended concept is unlikely to contain the negated word. This approach seems effective, although in some cases the assumption does not hold and a document deals with the intended meaning as well as contains the negated word. Such a document is relevant, yet not matched. If this approach is applied with care, it greatly improves precision, while only slightly decreasing recall. We learned this in the search tasks of Chapter 4.

The underlying assumption to this approach is that the more words occur in a document related to the unintended concepts, the less likely it is that the document deals with the intended concept of the designation. In reverse, the more words occur in a document related to the intended concept, the more likely it is that the document deals with the intended concept of the designation. We distinguish two cases, each based on one of these assumptions.

(1) Expanding a query by adding words related to the unintended concept results in a requirement of *absence* of all the related words in each matched document. If the list of related words is long, the expansion may exclude relevant documents. If the list is short, the expansion hardly affects the result of matching. The list will be relatively short if the domain representation is poor; in that case the expansion is ineffective, but also harmless.
(2) Expanding a query by adding words related to the intended concepts results in a requirement of *presence* of at least one of the related words in each matched document. In order to have all relevant documents matched, the list of added words must be large. Simultaneously, a list that is large has a reduced filtering effect. If the domain representation is poor, the list will be relatively short, and hence the recall will be diminished.

Dipe-D includes both kinds of adding words for improving precision. We only discuss the adding of words related to the unintended concepts; the other kind proceeds analogously.

Adding words can be done after matching, by relevance feedback. As an alternative approach, Dipe-D selects the words automatically, preceding the matching, and using the domain representation. If the domain representation adequately represents the domain, the effect will be similar to that of adding the words as a form of relevance feedback.

Dipe-D finds the designations related to the unintended meanings in the domain representation: these are the designations of concepts directly related to the unintended concepts of the ambiguous designation. For simplicity, the designations are only used literally. Moreover, only designations are selected that do not designate both concepts related to intended concepts and unintended concepts.

For example, assume that in the domain representation three meanings of "inheritance" exist: a legal, a biological, and one from knowledge representation in computer science. The related terms are:

legal	property, real estate, will;
biological	gene, chromosome, reproduction;
knowledge representation	type hierarchy, properties.

71 "meubel" means furniture.

If the legal meaning is intended, the query is disambiguated by adding:

"… ⋀ NOT ("gene" ⋁ "chromosome" ⋁ "reproduction") ⋀ NOT ("type hierarchy")"

The term "property" is not used, as it is related to both an intended and an unintended meaning.

We note that the above approach deals with ambiguous *designations* of concepts only. Although individual *words* in the resulting queries may be ambiguous as well, we do not disambiguate these words. The reason is that they are already disambiguated[72] by the presence of other words from the designation in the query, and the use of the proximity operator.

It is also noted that sub-queries are created individually. Interaction between the sub-queries is not taken into account. As a result, e.g., words may occur twice in a query. We do not expect that the ignorance of interaction between sub-queries affects the effectiveness of search; it does affect the efficiency, however. Modern computers are fast enough to make the latter hardly noticeable.

The algorithm in pseudo-code

Below, we present the algorithm for the transformation into a Boolean query of a collection of concept-ids, in pseudo-code. Collections are represented by variables starting with a capital. The addition sign stands for string concatenation.

```
procedure Boolean_query_to_concept_ids(prec_enhancement; recall_enh_type; Concept_ids; query);
begin
  D_ext_i := ∅
  for each c_i ∈ Concept_ids do begin
    find_designations(c_i, D_i);          {D_i is the collection of designations of c_i}
    for each d_j ∈ D_i do begin
      add d_j to D_ext_i;                 {D_ext_i is the extended collection of designations of c_i}
      if recall_enh_type = strong then begin
        add_quasi_synonyms(d_j, D_ext_i);
      endif;
    endfor;
    for each d_j ∈ D_ext_i do begin       {create sub-query to designation d_j}
      if (recall_enh_type = weak or recall_enh_type = strong) then begin
        select_words(d_j, W_ij);          {W_ij is the coll. of words to the designations d_j of concept c_i}
        W_ext_ij := ∅
        for each w_k ∈ W_ij do begin
          add w_k to W_ext_ij;            {W_ext_ij is the extended collection of W_ij}
          find_wordstem(w_k,ws);
          ws := ws + '*';                 {* is the truncation operator}
          add ws to W_ext_ij;
          find_lexical_variants(w_k,W_help);
          for each w_h ∈ W_help do add w_h to W_ext_ij;
          query_to_word(w_k) := ⋃ w;      (w ∈ W_ext_ij)
```

72 At least, as far as the designation itself is not ambiguous.

```
        endfor;
        query_to_designation(d_j) := NEAR_{w∈W_ext_ij} query_to_word(w_k);
      endif
      else begin {basic procedure}
       query_to_designation(d_j) := '"' + designation(d_j) + '"';
      endelse
      if prec_enh then begin
        find_designations_related_to_the_unintended_concepts(d_j, c_i,D_unintended);
        remove_ambiguous_designations(d_j, D_unintended);
        query_to_designation(d_j) := query_to_designation(d_j) +
                                     'AND NOT (' + ⋃_{d∈D_unintended} "d"+ ')';
      endif;
    endfor;
    query_to_concept(c_i) := ⋃_{d_j∈D_i} query_to_designation(d_j);
  endfor;
  query := ⋃_{c_i∈Concepts_needed} query_to_concept(c_i);
end
```

6.6.3 Transformation using partially representing concepts

The formulation of a query using partially representing concept identifiers is done by recursively calling the formulation process for fully representing concept identifiers. In each recursive call, a sub-query is made for one of the collections above; this is done according to Subsection 6.2.2, hence using the disjunction operator and the proximity operator. The total query then is constructed as the conjunction of the sub-queries; this means that a document is considered relevant only if it contains from each of the collections at least one concept. The query thus becomes:

(c11 OR c12 OR … OR c1k) AND … AND (cn1 OR n2 OR … OR cnm)

with the concepts replaced by their respective sub-queries according to Subsection 6.6.2 (for example, c11 may yield the sub-query "(machine NEAR part) OR component)".

Optionally the conjunction operator is replaced by the proximity operator, aiming at an increased precision. However, this effect is not obtained if the number of conjunctions between concept groups exceeds the distance between words the operator allows for.

The obtained query is called to be in the conjunctive normal form (Frakes and Beaza-Yates, 1992). Alternatively, we could have created the disjunctive normal form, by creating the tuples first, connecting the concepts in the tuples by conjunctions, and subsequently connecting the obtained groups by disjunctions: a document is considered relevant if it contains at least one tuple. Both methods of formulation yield the same documents.

In Subsection 6.6.2, we described the combination of sub-queries by the disjunction operator. There, the sub-queries indicated concepts needed, not describing concepts. The distinction between concepts needed and concepts describing concepts needed is not made in existing text retrieval systems. Yet, some of these systems offer the user an option for connecting the

designations in the query by either conjunctions or disjunctions (see, e.g., Roziç and Dimec, 1994; Notes, 1995).

We note that the use of partially representing concept identifiers (yielding a description of the information need) is expected to be less effective as the use of fully representing concept identifiers (yielding a direct designation of the information need). The reason is that an infinite number of descriptions is conceivable for each concept. Since only a few of these descriptions are covered by Dipe-D, a document is considered relevant only if it contains one of the descriptions covered; documents containing other descriptions or the designation of the concept itself are considered irrelevant.

The recursive formulation process may be applied even if the domain representation does contain the concepts needed; in such cases it supplements the formulation of a query with the concepts needed. The query then consists of two parts: one with the concepts needed, and one with the describing concepts. We expect that such a combination increases the recall; the increase will be marginal, since the effectiveness of using a description seems low. Simultaneously, the combination will increase the required amount of effort for query composition substantially. For these reasons, we have decided that the device supports using a description only if the concepts needed are not present in the domain representation (and, hence, making a description is the only option). By completing the formulation of a query, the query-formulation process is finished, and therewith the task of Dipe-D.

6.7 Chapter evaluation

The chapter is evaluated by a discussion (Subsection 6.7.1), a conclusion (Subsection 6.7.2), and three suggestions for future research (Subsection 6.7.3).

6.7.1 Discussion

The chapter has yielded the design of Dipe-D, a device for the formulation of an initial query. Dipe-D has a two-step approach, whereby in both steps a knowledge representation expressed in Dipe-R (see Chapter 5) is consulted.

An issue worth mentioning is that Dipe-D entirely depends on the consultation of an explicit knowledge representation. The latter is not necessarily based on the document collection. Hence, a conceptual gap may occur and deteriorate retrieval performance, as mentioned in Section 3.3. There are two reasons for this: (1) the proper concepts are not identified in the concept-identification step, and (2) the proper designations are not identified in the transformation step. Therefore, if the representation is not based on the document collection, relevance feedback should be used in order to improve the initial query (see Section 3.5).

6.7.2 Conclusion

The chapter assesses our third research question "*Is it possible to formulate a knowledge-based technique that effectively supports initial query formulation processes?*" As a result, it produced Dipe-D. The first step of Dipe-D's two-step approach consists of the development of the user's information need, and the expression of the information need by concept identifiers. This is realised in an iterative loop of question, answer, and explanation. The second step consists of the

automatic transformation of the described information need (i.e., the collection of concept identifiers) into a Boolean query. The transformation has variants including two recall enhancement techniques and a precision enhancement technique.

Dipe-D distinguishes itself from known approaches of query formulation by its two-step approach, and the way the two steps are performed. The first step is an alternative to concept identification by the known approaches, being explorers of graphical network presentations of domain knowledge. Both alternatives seem useful, each under different conditions. An advantage of Dipe-D over the known explorers is that it allows for the precise identification of a collection of concepts in a single expression, and for the exploration of the domain by subsequent expressions, varying from broad to narrow. In existing explorers, the exploration always happens relation by relation, and as a result may be less effective.

The second step shares the idea of expressing the information need as a collection of collections of concept identifiers with the query expansion technique by Järvelin *et al.* (1996). The second step differs in that it creates the query in a different manner (performing recall enhancement differently, and including a precision enhancement technique).

The techniques developed in Dipe-D are based on our findings about query formulation of Chapter 4, and the state of the art of query formulation described in Chapter 3. On the basis of this background, we believe that Dipe-D is a tool that supports the formulation of a query having a high retrieval effectiveness. The theoretical answer to our third research question therefore is affirmative. Of course, it still needs to be tested experimentally. This is done in Chapter 8.

6.7.3 Future research

Below, we provide three suggestions for future research towards the improvement of Dipe-D.

1. Support in the editing of an expression of the specification language

In the specification process, the user investigates his information need and explores the domain. This should be continued until the information need is sufficiently understood and expressed properly (i.e., the concepts meeting the expression are the concepts needed). In each repetition, the expression is revised. Revision is done by adding concepts, by deleting concepts, or by a combination thereof.

For adding concepts, the current expression should be relaxed. This can be realised by:
– deleting one or more sub-expressions,
– choosing a less strict sub-expression, and
– taking the designation of a supertype of types used in the sub-expressions.

For deleting concepts, a more specific expression should be made. This can be realised by:
– adding one or more sub-expressions,
– choosing a stricter sub-expression, and
– taking the designation of a subtype of types used in the sub-expressions.

If there are no concepts matching an expression, it obviously does not represent the user's information need. Assuming that the knowledge representation contains the concepts of the information need, it is clear that the user then needs a more general expression (i.e., one that is

met by a larger number of concepts).

In future versions, Dipe-D may support the revision of expressions. It may suggest changes to an expression, after the user has entered his suggestions: increase the number of concepts, decrease it, or change a concept. For this purpose, Dipe-D may use the ordering of sub-expression types as introduced in Section 4.2. The ordering is as follows, from strict to less strict: s1, s2, s3, {s4, s5, s7}, s6. The sub-expression types s4, s5 and s7 have equal strictness. Furthermore, any primed sub-expression type is less strict than its related sub-expression type, since it contains "any_type_of t", whereas its related sub-expression type contains "t" only. The primed sub-expression types cannot be directly compared to other (primed) sub-expression types. A sub-expression type with a letter extension is more strict than its related sub-expression type, since the former determines the direction of the relation in the sub-expression, whereas the latter ignores the direction.

For example, suppose that an expression contains sub-expression type 4 (requiring the concepts that are related to a given designation: **I_want_the_concepts_that are_related_to** disease. The sub-expression may be replaced by the stricter sub-expression type 2 (requiring the concepts that are related to the same designation, but by a given relation type); **I_want_the_concepts_that have_relation_of_type** treat **to** disease.

2. Supporting the association in the concept-identification step

In Chapter 4, the process of association was found of importance for developing the information need. Dipe-D's specification language offers some associative power, in particular by the sub-expression types 4, 5, and 7. An alternative way of associating is already mentioned in Section 6.2: the creation of several specifications using each other's output. Such support could be facilitated in future versions of Dipe-D, by allowing for the storage of pairs of the specification its and corresponding solution, as subsequently created in Dipe-D.

3. Improved expression of the information need

In Dipe-D, a collection of concepts expressing an information need contain concepts wanted and not wanted. In the future, the collection could distinguish concepts wanted absolutely from concepts wanted but not so much (i.e., "must" vs. "may"). Additionally, the concepts may be assigned weights, expressing how much they are wanted. This seems in particular useful for probabilistic matching.

Chapter 7

Dipe-D implemented and Dipe-R exemplified

The chapter reports on the implementation of Dipe-D and on an example knowledge representation in Dipe-R. Both are used in the experiments of chapter 8 for testing Dipe-D. Section 7.1 describes the implementation of Dipe-D. Section 7.2 has as subject the content of the example knowledge representation.

7.1 Implementation of Dipe-D

The section describes the implementation of Dipe-D. Chapter 6 provided the design of Dipe-D's main functions. For the actual implementation, the functions of Dipe-D as described in chapter 6 need to be worked out into more detail with some emphasis on the interfaces. This is done below. First, we briefly discuss the software tools used for the implementation (Subsection 7.1.1), then we summarise the functional architecture (Subsection 7.1.2), and report on the user interface design (Subsection 7.1.3).

7.1.1 Software

Below, we briefly discuss the tools used and how fast the processing is.

The tools

The software runs on a Windows95 platform. The functions of Dipe-D and Dipe-R are implemented using the programming environments Delphi (Version 2.0), and Amzi! PROLOG + Logic Server (Version 3.0). Delphi provides an object-oriented version of the programming language PASCAL and a toolbox for designing graphical user interfaces according to the Windows95 conventions. Amzi! PROLOG is a standard type (compiled) PROLOG for Windows. In the implementation, PROLOG and Delphi communicate by the Amzi! Logic Server, a bi-directional command interface.

As was already mentioned, the Delphi programming environment was used for the three main functions of Dipe-D and for a part of the interface of Dipe-R. This comprises approximately 6,000 lines of code. The remainder of Dipe-D (i.e., the definition of the subexpression types and

the interface rules to Dipe-R) and the interface of Dipe-R are written in PROLOG. This comprises some 200 lines. The retrieving and deriving are implemented entirely in PROLOG.

The processing speed

The implementation of Dipe-D itself does not yield problems with respect to processing speed. However, Dipe-D consults the example representation, and then prompts the latter to make derivations (in PROLOG). It is generally known that the derivation procedure of PROLOG sometimes yields impractical slow responses, and that infinite recursion may occur. In order to keep the response time of the representation low, and to prevent infinite recursion, we introduce the search depth as a variable in the PROLOG derivation rules. For each subexpression to be solved, the derivation process is cut off at a preset search depth. Search depth is defined as the maximum length of each branch of the derivation path. Thus, if a derivation rule has two conditions, each of the paths that merge in the rule is allowed to have a length equal to the 'search depth'. Not all derivations are counted in the search depth. For instance, the derivations made for translating a subexpression of Dipe-D to an expression of Dipe-R are not counted. Only derivations within Dipe-R count.

The limitation of the search depth may have as result that some of the solutions (concepts) of a specification are not identified. We empirically set the search depth such that in the example representation all concepts are identified[73]. The speed of the program was also promoted by empirically choosing an appropriate order of clauses in the PROLOG derivation rules.

Response times for solving a specification vary on Pentium 166 MHz computers with 32 MB of RAM memory from a few seconds to several minutes, depending on the peculiarities of the specification. For simple specifications, a response time may be expected below one minute, for elaborate specifications it may be two to four minutes. For our experimental purposes, we expect this to suffice. For practical application purposes, we suggest the use of faster computers.

7.1.2 Functional architecture

The functional architecture of the implementation of Dipe-D is depicted in Figure 7.1. The figure also shows the functional architecture of an implementation of Dipe-R; this implementation is consulted by the implemented Dipe-D. In the figure, boxes depict functions, and arrows represent communication. The italicised text in a box refers to the software tools used for the implementation of the function mentioned in the box. The functions performed by Dipe-D are categorised into four groups.

Dipe-D's functions:

1. user-interface functions
   a. creating, extending, and editing of specifications,
   b. presenting a solution of a specification,
   c. selecting concepts from a solution,
   d. exploring concepts in the solution,
   e. quick-browsing (see below),

73 When a different representation is used, the search depth needs re-adjustment.

2. solving specifications (by performing set operations),
3. solving sub-expressions, and
4. transforming a collection of concepts into a Boolean query.

Each function is implemented in a separate subprogram. The subprograms of Dipe-D communicate with each other as follows. The user interface calls the programs "solving expressions", and "transforming". The program "solving expressions" calls "solving sub-expressions". The user-interface program, and the programs "transforming" and "solving sub-expressions" call (the interface of) Dipe-R.

The two functions of Dipe-R are: (1) retrieving and deriving (of represented thoughts), and (2) interfacing to applications. They are implemented in two subprograms.

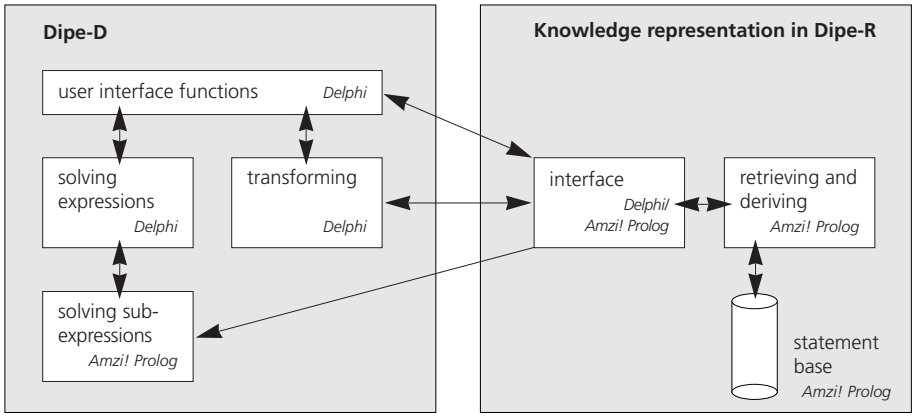

FIGURE 7.1: *Functional design.*

7.1.3 User-interface design

The user interface adopts the Windows 95 conventions. It comprises two main windows, and several supporting windows and dialogue boxes. The structure of the query-formulation process is essentially reflected in the window structure: each of the two main windows is the interface to one of the two steps of Dipe-D (concept identification and transformation, respectively). We describe the user interface by the main windows.

The concept-identification window

The concept-identification window (see Figure 7.2) contains two areas: one for the specification of an information need (specification area), and one for the solution to a specification (solution area). The specification area contains buttons for creating a new specification, editing a specification, extending a specification with a new sub-expression, and solving a specification. The concept-identification window supports the creation of a collection of concepts, the collection contains either fully representing concepts or partially representing concepts. The transformation window controls which option is used.

When creating, editing (i.e., changing and/or deleting sub-expressions) or extending a specifi-

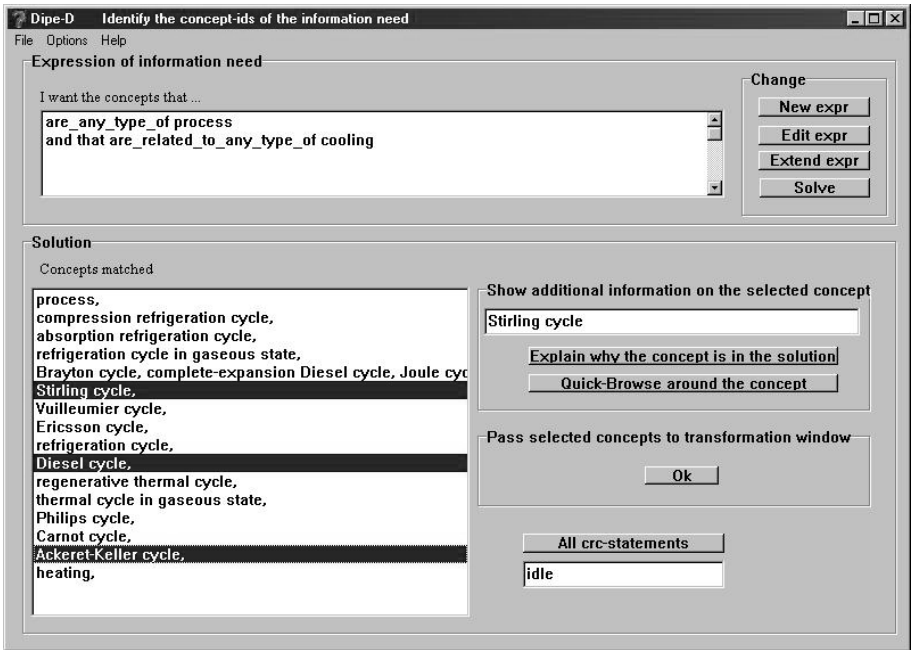

FIGURE 7.2: *Dipe-D's concept-identification window.*

cation, a listing of sub-expressions is shown in a dialogue box (see Figure 7.3). The sub-expressions are those developed in Chapter 5, supplemented with a sub-expression type for specifying the supertypes of a term. The user chooses one of two lists, a short list comprising the most essential sub-expression types, and a full list comprising all sub-expression types. The sub-expression can also be used in negated form, as described in Subsection 6.3.8. Once the user has selected a sub-expression, he has to fill out the variables in the sub-expression type selected (concept designations and/or relation type designations); this is done using dialogue boxes.

Once a specification is solved, the solution area of the first main window shows the concepts satisfying the specification. The user then should select concepts from the list: in Dipe-D selected concepts are shown marked (in Figure 7.1, three concepts are selected). On the command "pass", the selected concepts are passed on to the transformation window.

On the concept-identification window, two sub-windows can be opened: one for explaining the solution (called "explanation window"), and one for browsing around concepts (called "Quick Browser").

The explanation window shows how many times the concept satisfies the specification, and to which of these times the explanation shown corresponds. The explanation itself can be expanded and collapsed, hence varying the amount of detailing shown. The top line of the explanation shows the concept, the lower lines show the statements that together explain why the concept satisfies the specification. The explanation window only shows the content of a statement. An additional window can be opened for also showing its source information.

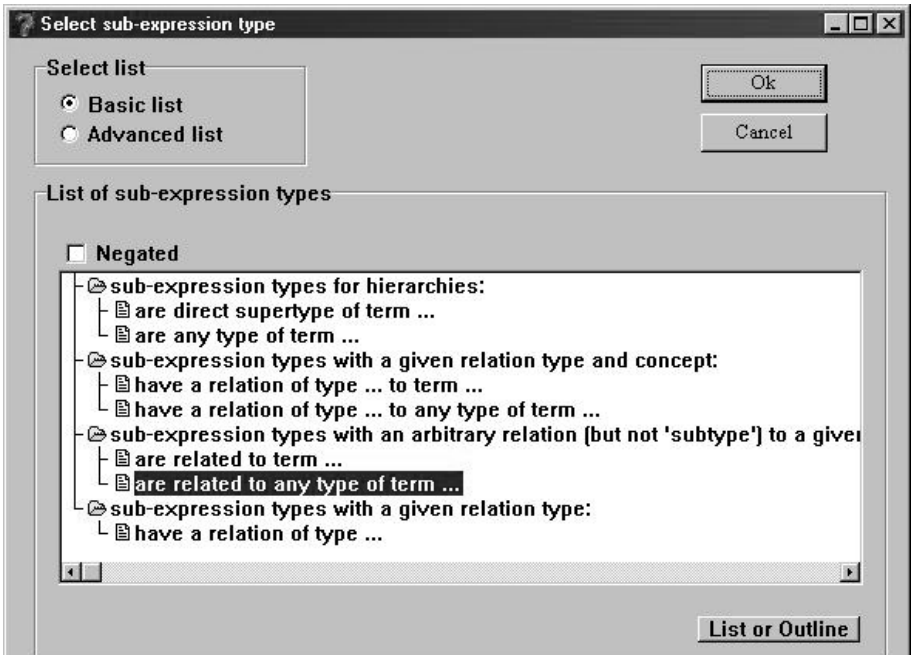

FIGURE 7.3: *Dipe-D's dialogue-box for selecting a sub-expression type.*

The Quick-Browser allows the user to ask for the concepts that are (a) supertypes of a given concept, (b) subtypes of a given concept, or (c) directly related to a given concept, or for (d) the statements containing a given concept.

The transformation window

Once the user has chosen the "pass" command in the concept-identification window, the window becomes invisible, and the transformation window becomes visible. Additionally, the concepts selected are passed on. Figure 7.4 shows the transformation window.

In the transformation window, there are three list boxes, in each of which the concepts of an individual specification can be stored. Usually, if the representation contains concepts that match the information need, only one box is used. However, if no such concepts are present in the representation, the information need may be described by two or three collections of concepts, as described in Section 6.2. The decision to use two or three collections – instead of only one – is prompted by the transformation window. We chose this window as the point of choice, and not the concept-identification window, since the former one encourages the user to try to create a single concept group first. He may introduce additional groups after he has failed to identify adequate single concepts for describing his information need.

Each of the groups of concepts can be edited or deleted. For the creation of new groups and for editing a group, the concept-identification window becomes visible again until concepts are identified, selected, and passed on to the transformation window.

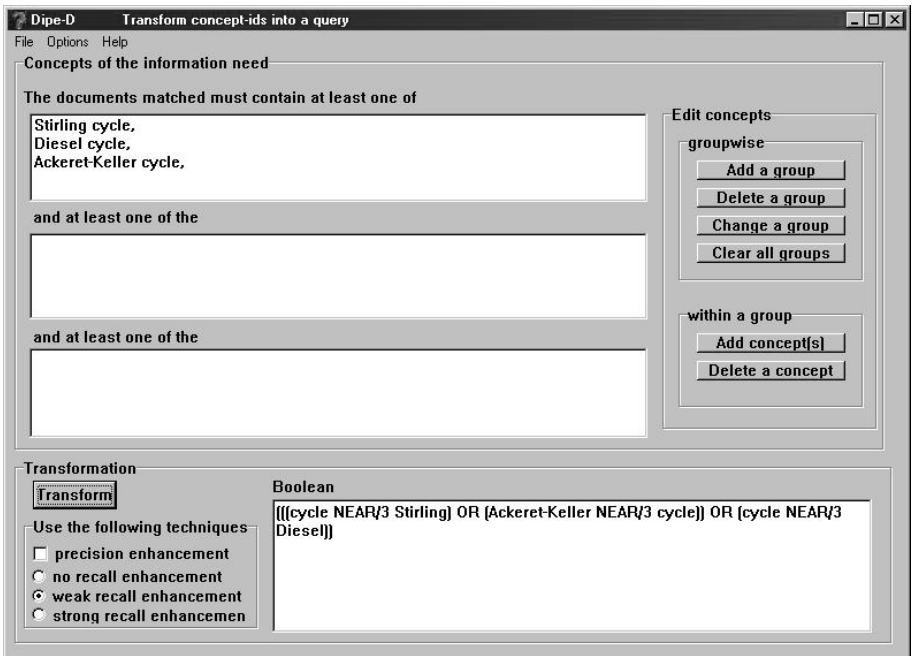

FIGURE 7.4: *The transformation window.*

Once the user has finished the selection of concepts, he commands Dipe-D to transform the collection(s) of concepts into a Boolean query. For this purpose, the transformation variants of Chapter 6 are implemented: basic transformation, and transformation using precision and/or recall enhancement techniques. Once the transformation is completed, the resulting query becomes visible. The query is then ready for matching by a retrieval system.

7.2 An example representation in Dipe-R

For the implementation of an example representation in Dipe-R, we choose the subject area of refrigeration (as an engineering discipline). In order to be able to represent subject knowledge, it is necessary to know what constitutes subject knowledge. We therefore first analyse the subject area of refrigeration: we introduce the area, describe the activities in the engineering process, and discuss what kinds of knowledge play a part (Subsection 7.2.1). On the basis of this analysis, we select the relation types and the accompanying concepts for the example representation (Subsection 7.2.2). Finally, we discuss some notable issues encountered during the representation task.

7.2.1 An example domain: refrigeration

Williams and Bobrow (1991) warn that an attempt to represent knowledge on a specific subject may easily end up in an attempt to represent enormous amounts of knowledge, dealing with the entire universe. Of course, this would take too much effort and would result in a large

amount of represented knowledge that is never used. In order to avoid this pitfall, Williams and Bobrow recommend representing only knowledge selected for a very narrow task. In our research, the specific task is that of supporting query formulation for searching documents containing design issues of equipment and processes of refrigeration (patent documents, as introduced in Section 4.2, are examples of such documents).

Background

Refrigeration[74] is a sub-domain of mechanical engineering, dealing with the processes and equipment for the removal of heat from substances. It includes processes and machinery for handling the latter as a stored product (PTO, 1996). Usually, the heat removal is realised through a process consisting of a change of phase of a coolant or a refrigerant. Such a process occurs by evaporation, melting or sublimation. Thus a central notion of refrigeration is the (forced, i.e., not spontaneous) transport of a particular kind of energy: heat.

The applications of refrigeration are domestic and industrial. An example refrigeration process is the compression-refrigeration process. This type of process is implemented in equipment consisting of a closed circuit of four components, through which circuit a fluid is forced to circulate (Stolk, 1990). The process is schematically depicted in Figure 7.5, showing a process diagram commonly used for depicting installations implementing thermal processes. The fluid in the evaporator evaporates, while taking up heat ($Q_{evaporator}$); it is then brought to a higher pressure in the compressor. Next, it condenses in the condenser, while giving off heat ($Q_{condenser}$). Finally, the fluid is given a lower pressure in the expansion valve, and returns to the evaporator.

The evaporation and condensation occur at equilibrium temperatures between liquid and gaseous fluid. With the fluids common for refrigeration purposes, the equilibrium temperature is proportional to the pressure. Thus, the evaporation temperature is below the condensation temperature. Hence, heat is transferred from a cold place to a warmer place, i.e., in the direction opposite to that in which it would spontaneously flow. This forced transfer requires the supply of energy to the fluid; it is done by the compressor. Typically, the compressor obtains the energy from an electric motor ($P_{compressor}$).

The whole loop satisfies the first law of (phenomenological) thermodynamics, an energy conservation law. Assuming that there are no energy losses (i.e., energy 'leaks') in the circuit, this law may be expressed as:

$$P_{compressor} + Q_{evaporator} - Q_{condenser} = 0.$$

The variables in this equation are:

$P_{compressor}$	the (mechanical) power supplied to the shaft of the compressor,
$Q_{evaporator}$	the (thermal) power supplied to the evaporator,
$Q_{condenser}$	the (thermal) power removed from the condenser.

74 The reason for choosing refrigeration is that the author has first-hand knowledge of this subject area.

In the engineering of a refrigeration loop, several mathematical expressions are used for calculating the powers of the various components, as functions of the thermodynamic state changes of the refrigerant in its cycling through the loop.

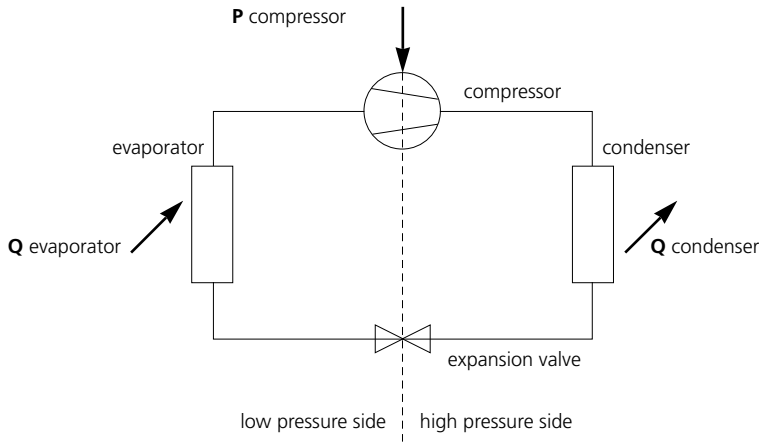

FIGURE 7.5: *Process diagram of compression-refrigeration process.*

Kinds of knowledge used in design tasks of refrigeration

For characterising the knowledge used in engineering tasks, we briefly discuss the contents of such tasks in refrigeration.

Engineering tasks in refrigeration aim at designing *devices,* and *processes* that satisfy a given collection of requirements. An example of a device is a piece of equipment used in a compression-refrigeration process as described above. Examples of a process are the compression-refrigeration process, and the handling of components in an assembly line of a refrigerator factory.

The primary requirements for the devices and processes is that they exhibit specific *behaviour*, under given circumstances (by behaviour we refer to changes in the material world). For example, a refrigerator usually is designed to have sufficient cooling power for the cold storage of goods in the envelope within a given temperature range, at given environmental conditions (i.e., given outside temperature, outside air humidity, and air velocity, etc.). Instead of behaviour, the notion of *function* is often used, to indicate (explicitly) intended behaviour. For instance, an evaporator has (by definition) as the function to evaporate the fluid it contains (thereby taking up heat from its environment). As a second example, a refrigerator has as main function the cold storage of goods.

According to Hamel (1990), an engineering process can be viewed as a problem-solving process. Hamel (1990) bases his problem-solving view of engineering on his observations of architectural engineering tasks. From our own experience in thermal engineering, it seems that this view can be safely generalised to engineering tasks in the latter area. A problem-solving process, according

to Newell and Simon (1972)[75], is defined as follows. A given problem is represented as a state in a state space. A solution is represented by another state in the state space, and the problem-solving process consists of creating at least one path (consisting of states and interconnecting steps) between the two states.

Below, we outline how the problem-solving process is usually done in engineering, based on our own experiences in thermal engineering. The process matches the above description, in which the problem is defined by a set of requirements, and the solution is defined by the design.

In general, an engineering process starts with an analysis of the requirements. In the next step, a design is conceived. In most cases it is not possible to deduce a design from the set of requirements straightforwardly. Instead, a design is conceived, and subsequently it is evaluated: will it perform the desired functions properly, and will it satisfy the other requirements? An essential part of the evaluation of each design consists of a prediction of the design's behaviour, i.e., of what changes an implemented design will make in the real world. In some cases (typically if the prediction contains a large amount of uncertainties), the prediction is validated by testing a prototype implementation.

If the design does not satisfy the requirements, a revised design is made. The evaluation then is also repeated. The conception (or the adaptation) of a design (in engineering) is done either from scratch, or on the basis of an old design with known behaviour (Eekels and Roozenburg, 1979). Thus, the engineering process iterates through subsequent stages.

On the basis of the (introductory) description above, we assume that in the engineering of devices in refrigeration, knowledge of the following seven kinds is used:

1. knowledge of the composition of the subject of the design itself,
2. knowledge of (legal) regulations for the design,
3. knowledge of the engineering process (e.g., its organisation),
4. knowledge of manufacturing costs of the design.

Additionally, three kinds of knowledge about the relationships between composition and behaviour are used:

5. laws of physics, i.e., the expressions of *causal* relations between individual events (a process is a series of events), obtained by induction over observed events in the physical world,
6. heuristic knowledge of the relation between composition and behaviour (similar to the knowledge of 4, but less fundamental), and
7. knowledge of composition and behaviour of specific pre-existing devices, i.e., the expressions of causal relations among events (dealing with specific instances, thus not generalised).

In all the seven kinds above, the world-view expressed in Section 2.1. seems that of most engineers. This viewpoint is usually implicitly assumed; it is seldom expressed explicitly.

---

75 See also Ollengren and Van den Herik (1989).

7.2.2 Selection of what is represented

The subsection describes what concepts and relation types are represented in the example domain, and provides some statistics on the representation.

Concepts and relation types

The example representation comprises only the information necessary for the task it supports: query formulation for searching design documents. Predominantly, engineering knowledge is knowledge of the composition of the subject of the design itself (type 1 mentioned above). We only express the 'prosaic' part of such knowledge, since Dipe-R does not support the representation of formulas, drawings, etc. Moreover, the example representation is too small for dealing with all the subjects of refrigeration. We choose to represent knowledge about a few subjects. These are:

1. basic concepts of thermal physics, e.g., energy, kinds of heat transfer, temperature,
2. cyclic thermal processes, e.g., the Stirling cycle, and
3. devices implementing thermal processes, in particular heat pipes.

The subject knowledge was obtained from the following sources: Diktaat Koudetechniek A1 (Stolk, 1990), Dunn and Reay (1994), and Dictionary of Engineering (McGraw-Hill, 1997). These knowledge sources were supplemented with the author's knowledge of the subjects. All the concepts represented are generic concepts.

Of course, the pre-represented concepts and relation types of Dipe-R, as enumerated in the appendix A, are also present in Dipe-R. In addition, we represent the concept 'stuff' as super-type for the concepts that do not refer to concepts (in general), relation types and symbols. The concept 'stuff' serves to refer to all kinds of things in the world (outside the brain), such as objects, fluids, energy, and matter. Borst, Top, and Akkermans (1994) inspire this concept.

Of the relation types, the pre-represented relation types are present and additionally we represent 16 relation types. They are listed below, with the accompanying concepts serving to define the roles of the relation types. For a better readability, each relation type is specified by its designation, and the designations of its two roles – hence, not by a number. In the implementation, numbers are used instead as concept identifiers (as was described in Chapter 5). The relation types are:

Relation types for the first sentence type of Dipe-R:
(concept) was_developed_for_function (event) …
(concept) was_developed_for_application_in (concept) …
(concept) is_somehow_related_to (concept) …
(concept) was_developed_for_improved_performance_with_respect_to (concept) …
(concept) was_developed_to_have_feature (concept) …

Relation types for the second sentence type of Dipe-R:
(concept) … are_caused_by (concept) …
(event) … occurs_in (state) …
(event) … occurs_with (event) …
(matter) … has_state (state) …
(event) … starts_in (state) …
(event) … ends_in (state) …
(stuff) … comprises (event) …
(event) … comprises (event) …
(stuff) - contains (stuff)
(concept) were_applied_in/for (concept) …

As already mentioned in Chapter 5, each concept in the representation is declared a subtype of one of the concepts above. This is necessary in order to be related to other concepts by relations of the types above (or the prerepresented relation types).

Statistics

Figure 7.6 provides statistical information about the example representation.

Total number of thoughts represented, for the representation of:	2,300 (approx.)
Type hierarchy of concepts (subtype)	1,075 (approx.)
Features of stuff concepts (excluding subtype-relations) (real)	178
Features of stuff concepts (noise)	278
Features of relation-type concepts:	
Role definitions of relation types	38
Transitivity of relation types	6
Quasi-designations	82
Lexical variants	32
In these thoughts there are approx. concepts	1,219
Stuff concepts	600 (approx.)
Symbol concepts	600 (approx.)
Relation type concepts	19

FIGURE 7.6: *Statistical information on the content of the example representation.*

We make three remarks to Figure 7.6. First, in type hierarchies each feature is represented in the top concept having the feature. In the lower concepts (subtypes), the feature is obtained by inheritance. As such, the number of explicitly represented features (178 + 278 = 456) is smaller than the number present in the representation, including derived features.

Second, the 278 noise features are arbitrary yet plausible features. For example, they relate

heat and cold. They are meant to enlarge the knowledge representation in an artificial manner.

Third, some concepts have several features. This means that other concepts can be conceived that have only a subset of the features. Since the total number of subsets of n features equals 2^n (1, the number increases fast with n. Representing a concept for each conceivable subset of features thus leads to many concepts. We take a different approach: we only introduce a concept if it is really encountered in the subject area; the possibility of existence, resulting from a combination of features alone, does not suffice for representation.

7.2.3 Discussion and conclusion

We discuss the representation of knowledge in Dipe-R, in particular its lack of expressive power. Then we provide a conclusion.

Limitations of expressive power

In Dipe-R it is not possible to express that an enumeration of features is exhaustive. Thus, it is not possible to indicate that the represented subtypes of a concept are precisely all subtypes. This plays a role in expressing the common knowledge that there are precisely three types of heat transfer processes: (1) thermal convection, (2) thermal conduction, and (3) thermal radiation. They are represented as subtypes of heat transfer. Thus, in the representation it cannot be distinguished whether it is generally accepted that there are three types of heat transfer, or that there were only three types mentioned and there may exist even more types. In line with the above, it is generally accepted that the total *amount* of heat transfer is the sum of heat transfer by the three mechanisms. This cannot be represented either.

Dipe-R cannot handle amounts. Energy conversion is the conversion of (an amount) of energy from one type of energy into another. The amount is not represented.

Dipe-R cannot handle alternatives: this *or* that is the case. For example, some phase-changes start or end in one of two states: evaporisation ends in vapour or gas. This is represented by two statements: "some instances of (event) evaporisation end_in (state) vapour", and "some instances of (event) evaporisation end_in (state) gas phase". The statements together do not have precisely the same meaning as the initial statement, since it is not possible to express disjunctions in Dipe-R.

Dipe-R cannot express similarities, or identity. Thus, if two subtypes of a concept have almost identical properties, these properties need to be enumerated twice. This lack of possibilities plays a role in the representation of the Stirling cycle and the Philips cycle. It would require fewer statements if it could be expressed that they have identical features, barring one (and also indicating which one).

Dipe-R cannot represent that a statement is made provisionary. The Philips cycle is represented as a subtype of the Stirling cycle. We know differences between the two cycles, but we do not know (yet) which of the differences are defining and which are coincidental. This cannot be expressed in Dipe-R.

We have represented "some instances of heat pipe have_function an instance of thermal switch". Moreover, we introduced the concept of "thermal switch heat pipe". Dipe-R does not enable the representation of the obvious conclusion: "some instances of heat pipe are an instance of thermal switch".

Conclusion

Although the example representation is created in support of experiments on Dipe-D, it also allows for an evaluation of Dipe-R. After creating the example representation, we draw the conclusion that it expresses a fair amount of domain knowledge, enabling query keys to be identified by the hierarchical positions and features of the concepts they designate. It thus meets the second requirement of Dipe-R (see Section 5.1) in practice. We did not evaluate whether the example representation also meets the first requirement (appropriate expressions). In Chapter 8, in experiments on Dipe-D, a start of such an evaluation is made.

Chapter 8

Experimental support for Dipe-D

The chapter reports on experiments for exploring the effectiveness and the weaknesses of Dipe-D, or otherwise stated the effectiveness and the weaknesses of our two-step query-formulation approach substantiated in Dipe-D (see Chapter 6). Two experiments are reported, one on each of the two steps of the query formulation approach. Additionally, Dipe-D is compared to ANS, and Dipe-R is evaluated briefly. The starting points for the chapter are provided in Section 8.1.

The first experiment serves to explore the effectiveness and the weaknesses of Dipe-D's concept-identification step. In the experiment, test persons use Dipe-D for identifying the concepts of a given information need. The experiment is reported in the Sections 8.2 through 8.4. These sections provide the design, the results, and conclusions, respectively.

The second experiment serves to explore the effectiveness and the weaknesses of Dipe-D's transformation step. In the experiment, Dipe-D transforms given collections of concept identifiers (representing information needs) into Boolean queries. Test persons do the same. The resulting queries are matched in a retrieval system. The documents thus matched are evaluated for relevance with respect to the given information needs. The effectiveness of matching by Dipe-D and by the test persons is then compared. The experiment is reported in the Sections 8.5 through 8.7. These sections provide the design, the results, and conclusions, respectively.

The comparison of Dipe-D to ANS is based on the findings in the experiments for concept identification, and is reported in Section 8.8. The evaluation of Dipe-R is based on the two experiments and a small additional experiment, and is reported in Section 8.9. Suggestions for future research on Dipe-D are provided in Section 8.10.

8.1 Starting points for the experiments

As we already mentioned, the goal of the chapter is to test the effectiveness of Dipe-D. First, we motivate our choice of an important variable in the experiments: the knowledge representation (Subsection 8.1.1). Then, we outline the experiments (Subsection 8.1.2).

8.1.1 Choice of the knowledge representation in the experiments

By what experiments can the effectiveness of Dipe-D be tested? Ideally, we would do various experiments in which all the influential variables are varied, like the knowledge representation (determining the subject area and the vocabulary), the type of task, the type of test person, etc. We thus would learn about the performance for many combinations of values. In practice, only a subset of such combinations is feasible.

The values of the variables in our experiments are selected in the sections that detail the experimental set-ups, viz. Sections 8.2 and 8.5. In the present section, we discuss only one of the test variables: the knowledge representation. For this variable the selection of the value is more or less given, for the following reason. As was explained in Chapter 6, Dipe-D operates by consulting a suitable knowledge representation. Thus, for an experiment in query formulation with Dipe-D there needs to be a knowledge representation covering the subject areas of the information needs in the experiment. As we have argued in Chapter 5, we did not find an existing, appropriate knowledge representation for Dipe-D.

We therefore created a representation ourselves: the example representation of Chapter 7. This representation contains approximately 600 represented concepts, in which all concepts have designations. Most concepts have features. The representation contains only a few synonymous terms, quasi-synonymous terms, spelling variants and homonyms. The example representation is small for the purpose of testing, but the creation of a larger knowledge representation lies outside the scope of the thesis.

Despite the limitations set by its size, the example representation enables useful tests. The number of concepts is sufficiently large as for the user to be unable to recognise the proper concepts in a listing of concepts, or in a listing of all the statements in the representation. Moreover, the number of features seems sufficient to be able to identify concepts by their features. The designations of the concepts allow for making queries. As such, the example knowledge representation allows for testing the two steps of Dipe-D: concept identification and transformation.

Although the example representation is suited for elementary tests, it does not allow for testing the ultimate merits of Dipe-D. This can be seen as follows. Regarding the concept-identification step, the four conditions in which Dipe-D should outperform other approaches cannot be fully exploited[76]. In the first place, the network is not dense. Second, the information need may comprise several features, but the knowledge representation does not always contain all those features, since the number of features represented is not large. Third, the features represented are very specific, and thus inheritance will hardly occur. So there are not several levels of abstraction represented. Fourth, there are not *many* concepts sharing identical features, although there are some.

Regarding the transformation step of Dipe-D, the presence of only a few alternative designations may be regarded not to constitute a problem, since there are only few variants in the subject area. Yet, this will allow for a simple test of the transformation step and its recall

76 The conditions are mentioned in Section 6.2: (1) the network of the representation is dense, i.e., has a high number of relations per concept, (2) the information need comprises concepts each having several features, (3) the user does not know the features of the concepts with the proper level of abstraction, and (4) the information need comprises several concepts having identical features.

enhancement technique. The absence of homonymous terms is due to the fact that there is only a single subject area represented and thus only a single concept represented for each designation. Thus, the precision enhancement technique (utilising homonymous designations) cannot be tested.

In summary, the example knowledge representation allows for testing, in a rudimentary sense, whether the working principles of the first and second step of query formulation in Dipe-D are effective at all. The knowledge representation does not allow for tests of the more subtle merits of Dipe-D.

8.1.2 Outline of the experiments

Dipe-D is tested seperately, in the sense that we do not perform experiments in which it is compared to other approaches for query formulation. The reasons for this are twofold. First, there is no other tool integrally supporting the initial query-formulation process. A comparison needs to be limited to a comparison of one of the two steps of Dipe-D, since alternatives exist only for each step (see Chapter 6). The second reason for omitting comparative experiments is that our example knowledge representation seems too small to show the precise differences among Dipe-D and other tools.

In spite of these restrictions, we believe that these experiments give a fair impression of Dipe-D's effectiveness and its weaknesses. The two experiments are as follows.

Test of the concept-identification step

The first experiment serves to test the effectiveness of Dipe-D's concept-identification step. In the experiment, six test persons each perform three search tasks. In each task, concepts are described in natural language. The concepts need to be identified by the user, with the aid of Dipe-D and the example knowledge representation described in Chapter 7.

Test of the transformation step

The second experiment serves to test the effectiveness of Dipe-D's transformation step. Eight test persons each transform by hand three given collections of concepts (representing an information need) into three corresponding Boolean queries. Dipe-D does the same. Then, the queries are matched and the documents identified are judged for relevance with respect to the information needs. The performances of the test persons and Dipe-D are compared, yielding an impression of the effectiveness of Dipe-D's transformation step.

Besides knowing whether Dipe-D is effective, we would like to be able to position Dipe-D relative to other approaches. Due to the absence of a larger knowledge representation, the benefits of Dipe-D over those approaches will probably not show, and therefore seems not useful. Yet, in order to obtain an impression of the relative performance of Dipe-D, we compare Dipe-D's concept-identification step to concept identification by the ANS browser (see Section 6.2 for an introduction to the working principle of this browser). The comparison is based on an analysis of the experiments of Dipe-D as reported in Sections 8.2 through 8.4.

Although the two experiments focus on testing Dipe-D, they also provide information on Dipe-R. We use this information for a preliminary evaluation of Dipe-R. We evaluate whether it meets the two requirements on which its design is based (see Section 5.1), how useful the relation

type information in Dipe-R is, and finally whether Dipe-R adequately supports query formulation. For evaluating the relation type information, a small experiment is performed in which we compare a simple list browser, having no relation type information, to Dipe-D.

The two experiments, the comparison of Dipe-D to ANS, and the evaluation of Dipe-R are worked out in the remainder of the chapter.

8.2 Design of experiment for the concept-identification step

The section consists of seven subsections, presenting the design of the experiment for the concept-identification step. Subsection 8.2.1 formulates two hypotheses. Subsections 8.2.2 - 8.2.5 provide features of the experiment. Subsection 8.2.6 describes what information is collected in the experiment.

8.2.1 Two hypotheses

The objective of the first experiment is to test the effectiveness of the concept-identification step of Dipe-D. According to Chapter 6, the concept-identification step supports the intertwined development and expression of the information need and as such it supports the identification of the concepts of the information need.

In the experiments we provide test persons with search tasks, i.e., concept-identification tasks. These tasks thus are performed in a system comprising Dipe-D, a test person, and a knowledge representation in Dipe-R. We consider the user as a given item in this system, i.e., not as a variable to be tested. All performance less than 100% thus can be ascribed to Dipe-D and the knowledge representation. It is not obvious that Dipe-D's performance will be 100%, even when the knowledge representation provides adequate support. Several problems may prevent such a performance. For instance, the test person may not understand the specification language, or the explanation. Moreover, the specification language must not be too awkward for specifying the information need in practice[77].

The effectiveness of the concept-identification step is measured by two variables, in analogy to the precision and recall of the matching of a query: concept precision and concept recall. We define these concepts as:

$$\textit{concept precision} = \frac{\textit{the number of concepts identified and relevant}}{\textit{the number of concepts identified}}$$

$$\textit{concept recall} = \frac{\textit{the number of concepts relevant and identified}}{\textit{the number of concepts relevant in the KR}}$$

On the basis of the above considerations, we formulate two hypotheses, each directed at one of the two above variables. We realise that these are rather bold hypotheses; they do not allow for any imperfection in Dipe-D.

77 Solving a specification, by Dipe-D, should not fail, since we already tested that in the implementation stage (see Chapter 7).

Hypothesis 1

Dipe-D supports the user in developing an information need from a description by features in natural language into a collection of concepts with a concept precision of 100%.

Hypothesis 2

Dipe-D supports the user in developing an information need from a description by features in natural language into a collection of concepts with a concept recall of 100%.

8.2.2 Outline of experiment

As already mentioned, the experiment comprises concept-identification tasks. In each task, a person is given a (natural-language) description of an information need, and must identify the relevant concepts of that information need using Dipe-D. The latter consults the example representation in Dipe-R (see Chapter 7). The subject area of the information needs is that of refrigeration. In each task, it is first verified that the test person does not know the answer by heart, and does not recognise the answer in a listing of all concepts in the representation (at least not in a short period of time). Thus, it is verified that without Dipe-D the relevant concepts are not identified. Hence, if the concepts are identified using Dipe-D, this can be ascribed to Dipe-D.

Two variables in the experiment are the number of test persons, and the number of search tasks. Of course, the larger the numbers of each, the more we learn from the experiments. It lies outside the range of this thesis to do experiments at a large scale; instead, we choose three tasks, each done by six test persons. This seems sufficient for obtaining a fair impression of the typical merits and problems of Dipe-D, in a qualitative way.

Each test person completes the tasks in a single session. The sessions utilise specific equipment (e.g., the implementation of Dipe-D), and take place in a specific environment. Moreover, in all sessions the same procedure is followed.

Below, we elaborate on the experiments: the environment and equipment (Subsection 8.2.3), the session procedure (Subsection 8.2.4), the search tasks (Subsection 8.2.5), the test persons (Subsection 8.2.6).

8.2.3 Environment and equipment

The equipment used in the experiments and the environment in which the experiments are performed may influence the results of the experiments. Therefore, they are described below.

Environment

All experiments are performed on three locations, both offering a room resembling an average office-room, with respect to quietness and comfort. In each experiment, only the test supervisor and one test person are present. The test supervisor takes notes, meanwhile maintaining a distance of a few meters from the test person and not interfering with the experiments in other ways than by providing instructions and explanation. We recognise that the mere presence of a test supervisor may have an effect on the test person's performance.

Equipment

In all the experiments, Dipe-D and the example KR run on similar computer equipment: Personal Computers having Pentium processors. Due to the fact that the experiments are performed on

several locations, various computers are used; they differ somewhat in processing speed, due to differences in clock frequencies and sizes of random access memory (RAM). Clock frequencies vary between 100 and 200 MHz; RAM varies from 16 to 32 MB. In preparations of the experiments, we found that even the slowest of these computers still provides an answer to most specifications in approximately one to two minutes (see also Chapter 7). Although this is not a convenient speed, it is considered acceptable for the experiments.

8.2.4 Session procedure

In each session, one test person completes three search tasks (and also two search tasks for the additional experiment described in Section 8.8). In the session, only the author is present, having the roles of observer and test supervisor. The following procedure is obeyed:

– the test person receives a handout (printed in the Appendix C); this includes an introduction to Dipe-D, an example task, the five search tasks mentioned above, and questions,
– the test person receives an introduction to Dipe-D, on the basis of the example task in the handout,
– the test person completes the two tasks for the additional experiment,
– the test person completes the three tasks for the concept-identification experiment,
– the test person answers the questions on the instruction sheet.

In his role as observer, the author observes the test person, and takes notes. The computer records information on the tasks and writes this information into a file (henceforth "logfile") – see Subsection 8.2.7.

In his role of test supervisor, the author provides an explanation about Dipe-D during the example task. In this instruction, each test person is advised to use the iterative loop as follows: start by a vague specification of the information need, and subsequently create more precise specifications – on the basis of the answers given and their explanation. Additionally, the test person is advised to use the quick-browser part of Dipe-D (see Chapter 6), for exploring what concepts are directly related to a specific concept identified.

The role of test supervisor further comprises the provision of more explanation of Dipe-D, when asked, during the auxiliary experiment and during the first task of the main experiment. The explanation is restricted to the way the iterative loop can be used; there is no support on the content of the tasks nor on what specific action is best at the specific point in the task at hand. During the second and third tasks of the latter experiment no information is provided at all.

8.2.5 Search tasks and solutions

The subsection discusses the search tasks and their solutions.

Search tasks

In each task, the test person is provided with an information need expressed in natural language, describing the concepts in features and types. These tasks resemble how real information needs are formulated, when they are communicated (see Chapter 4), at least in patent search tasks.

The test person is asked to identify the relevant concept-identifiers using Dipe-D. As already mentioned, the tasks are chosen such that the test persons cannot readily provide the answers. The amount of time a person is allowed for each of the three tasks is approximately 30 minutes

(including the processing time of Dipe-D). In preliminary experiments this proved sufficient, but not abundant.

The subject area is that of the example knowledge representation, i.e., refrigeration (thermal engineering). The tasks are shown in the Appendix C, which also shows the two tasks for the additional experiment of Section 8.8.

Solutions

The solutions to the tasks are provided by the creator of the example knowledge representation, on the basis of his ready knowledge and from the books on which also the knowledge representation is based. The solutions are given in the Appendix D.1.

8.2.6 Test persons

The test persons must have the appropriate knowledge to understand the search tasks. Simultaneously, they must not have the knowledge to generate the answers by heart. We have tried to satisfy these conditions by selecting test persons by their knowledge and experience.

Our analysis of knowledge and information in document-search processes (see Section 4.1) shows the types of knowledge needed in the first step of query formulation. These are: knowledge of the work task, and the work task domain. Additionally, knowledge of the equipment used is needed, and knowledge for communicating: to the equipment, to the test supervisor, and for reading the handout.

We assume that this knowledge is present by selecting test persons having the following characteristics[78]:
– a technical education at university level, including some (phenomenological) thermodynamics,
– some experience with Windows95,
– a fair command of the English language.

The levels of knowledge relevant to the search tasks are verified on the basis of questions posed before the person starts executing the search tasks. See the Appendix C.

8.2.7 Information collected

From the experiments, information is collected for determining the results. We collected information of the following three types.

First, we need to know the concepts identified, in order to determine the effectiveness and thus to evaluate the hypotheses. Second, we verify the knowledge levels of the test persons, as described in Subsection 8.2.6. Third, we need additional information for being able to explain (unexpected) low effectiveness or other unexpected events occurring in the search tasks.

The information mentioned is collected as follows:
– collecting the answers to the questions on the handout,
– creating a log-file of the communication between each test person and Dipe-D,
– taking notes, thus describing the acts of the test persons, and the remarks they make.

78 The six test persons are all males; this is accidental, i.e., not on purpose.

Each communication between a test person and Dipe-D is recorded as a quadruple of
– communication type,
– time-tag,
– input message, and
– output message.

8.3 Results for the concept-identification step

The results of the experiments are the logs and the answers given to the questionnaire. In the Appendix D.2, a summary is presented of the completion of one search task in the main experiment is presented. The concepts identified in reply to the questionnaire are shown in the Appendix D.3.

From the concepts identified, we derive numbers expressing the concept precision and concept recall. On the basis of these results, we derived the values of the concept precision and the concept recall. They are reported in the Tables 8.1 - 8.4, below:

Concept recall per test person per task (Table 8.1),
Concept precision per test person per task (Table 8.2),
Averages of concept recall and precision, per task (Table 8.3),
Averages of concept recall and precision, per test person (Table 8.4),
Averages over test persons and over all tasks (Table 8.5).

Person (#)	Task 1	Task 2	Task 3
1	100	100	100
2	100	100	66.7
3	20	100	33.3
4	60	100	33.3
5	60	100	33.3
6	60	100	33.3

TABLE 8.1: *Concept recall (%), per combination of test person and task.*

Person (#)	Task 1	Task 2	Task 3
1	100	100	80
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100
6	100	100	100

TABLE 8.2: *Concept precision (%), per combination of test person and task.*

Task (#)	1	2	3
Average concept recall (%)	67	100	50
Average concept precision (%)	100	100	97

TABLE 8.3: *Average concept recall and average concept precision, per task.*

Test person	1	2	3	4	5	6
Concept recall (%)	100	88.9	53	65	69.4	69.4
Concept precision (%)	93	100	100	100	100	100

TABLE 8.4: *Average concept recall and average concept precision, per test person.*

	overall average
Concept recall (%)	74
Concept precision (%)	100

TABLE 8.5: *Overall averages of concept recall and concept precision.*

Below, we mention the main findings from the logs and observations made during the search sessions.

Specification

We discuss two aspects of the specification: the pattern and the subexpression types used.

1. Specification pattern

The questionnaire recommends to the test persons a pattern of specification in the iterative loop of specification, solving, and explaining. It is the pattern according to the theory of Chapter 6, i.e., starting by a broad specification, and making it more specific on the basis of the results. In the logs it is visible that the test persons adopt this pattern. Moreover, they use it several times within a single search task: if after refinement the solving of an expression yields the empty set, then a test person restarts by a broad specification (now with an entrance different from the former) and restarts the successive refinement.

2. Subexpression types used

In the experiments, only a few different subexpression types of the specification language are used. The subexpression types used are the subexpression types s1', s2, s2', s4, s4' (see Chapter 6).

Solving

The procedure for solving an expression of the specification language (as described in Chapter 6 and Section 7.1) has a search depth that is set each time the program of Dipe-D is started. In one experiment (that of test person 4) the search depth was set to a smaller depth, in order to

reduce the search time. The search depth was two instead of three[79]. This search depth resulted in the absence of one concept of the solution proper. As a result only two concept identifiers were found, and not the identifier of a sister concept (see the Appendix D.3). Since in all the other experiments the three concepts were identified simultaneously and recognised simultaneously by the test persons, we assume that with a proper search depth the test person would have identified the third concept as well. Hence, we count it in.

Explanation

The explanation function was used frequently. It was not used after an expression containing only a subexpression of the type "… are any type of …"; the answer then was obvious to the test persons. It was used almost always after a more elaborate expression, i.e., an expression of several subexpressions.

Within the explanation function, it is explained per concept why it matches each of the subexpressions. The test persons inspected the explanation only partly; the subexpressions of the above-mentioned subtypes were not inspected.

In the subexpressions of the type "… have a relation of type <r> to <t>" and "… have a relation of type <r> to any type of <t>" the explanation is sometimes inspected. The test persons then wanted to know the source of the statement that satisfied the subexpression.

In the subexpressions of the type "… have a relation to <t>" and "… have a relation to any type of <t>" the explanation is often inspected. According to the comments of the test persons, the inspection was needed for learning what the relation type is in the statements that satisfy the subexpression.

Quick Browser

The Quick Browser was used incidentally, usually after inspection of the explanation and when the solution was the empty set (in order to find better terms for a new specification). The latter occurred several times in the third task. On the basis of the findings in the Quick Browser then a new specification was made, or it was decided to not search any further.

Questionnaire

The answers to the questionnaire in the handout yield the following findings:

– The test persons answer that the subexpression types used in the expressions are clear.
– The test persons answer that the subexpression types used offer sufficient expressive power.
– The test persons answer that the Quick Browser is sometimes helpful for finding additional domain information to the explanation.
– The test persons answer that they understand the explanation function, in general. They mention two disadvantages:
  1. There is too much information in the explanation. Usually, in these experiments, the explanation is only used for learning a relation type – as mentioned above. The other

79 We note that the number of derivations *per sub-expression* is usually much higher than the search depth, due to branching. For example, a search depth of three will search (by the derivation rule for transitivity) six steps deep in a type hierarchy. For an introduction to the notion of search depth, see Subsection 7.1.1.

information is regarded confusing. Two test persons mention this in reply to the questionnaire. It is also observed by the test supervisor, on the basis of the way the test persons inspect the explanation screen and by their remarks during the sessions.

2. The thought types of the interface between Dipe-R and Dipe-D are not understood. All test persons recognise them as unimportant in-between steps of the explanation of how a subexpression is met by represented thoughts, and therefore ignore them.

– The test persons state that they believe having identified all the relevant concept identifiers, as far as they are present in the example representation. For the degree of this belief they answer "confident", in a series of alternatives: very unconfident, unconfident, neutral, confident, and very confident.

8.4 Evaluation

The evaluation is divided into three subsections: a discussion of the results (Subsection 8.4.1), conclusions (Subsection 8.4.2), and suggestions for future research (Subsection 8.4.3).

8.4.1 Discussion

We discuss the results in the light of our two hypotheses. We apologise that the text becomes quite technical (in the domain of thermodynamics) and therefore more difficult to read (for computer scientists) then so far. However, we believe that in order to communicate some of the results we should have part of the results in our main text and not all in the appendices.

Hypothesis 1: concept precision

All relevant concept-identifiers are identified (6 * (5 + 3 + 3) = 66) except one. For reasons mentioned above (search depth), we ignore this single exception.

In the first task, the expert judge considers the open and closed Brayton cycle as solutions, as well as the Brayton cycle sec. The reason for this is that (until today) all Brayton cycles are either open or closed (see the Appendix D). In the third task, the expert judge regards three concepts as solutions, but not their common supertype "heat pipe". The reason for this is that there are many types of heat pipe, and not only the three specific types that meet our task.

One of the test persons gives both in the first and the third task as answers on the common supertype (Brayton cycle, and heat pipe). In the third task this is wrong, but Dipe-D offers this answer when the proper features are given; namely, via the noise ("is somehow related to"). The test person did not check the explanation facility, in which he could have seen that the relation is vague and therefore probably wrong. As a result, he selects one superfluous concept (four instead of three). We note that this task by this test person has the highest recall of all the test persons for this task. He is the only one that has identified all three relevant concepts (plus one irrelevant concept). It thus may be that his focus was on concept recall rather than concept precision.

Hypothesis two: concept recall

The concept recall is significantly lower than 100%. Below, we analyse the results (task by task) in order to explain why the recall is not 100%.

Task 1

The first two test persons each identify all (five) relevant concepts. The third test person initially has the right approach of refining, and in the solutions of one of his specifications the relevant concepts are precisely identified. However, he then adds another feature and loses the solution. He adds the feature because it is a feature asked for in the task. It seems in itself a good approach to verify the satisfaction of all the features by the concepts identified. Unfortunately, the last feature specified is not represented in the same terms in the task as in the knowledge representation. The test person does not try other ways of expressing the last feature and therefore identifies only one of the five relevant concepts. The problem occurring is that the feature is not expressed in the specification in the same manner as it is expressed in the knowledge representation. Apparently, it is difficult for the test person to find the proper expression.

The fourth and fifth test persons do not identify the two types of Brayton cycle: open and closed. The Brayton cycle is identified by a specification containing a feature represented by a relation type that does not inherit (viz. the relation type "was meant to be used in …"). As a result, the feature is expressed only for the supertype (Brayton cycle) and not for its subtypes. Having identified one relevant concept, the test person has no idea that he can identify more relevant concepts by using different features.

Task 2

In task 2, all the relevant concepts were identified.

Task 3

Task 3 is the most troublesome task for the test persons. In the log files we find two causes for this. The first cause is that of translation. If the given features and types in the task are literally used in a specification, no concepts are identified at all or no relevant concepts are identified. They have to be expressed differently from how they are provided in the task. It appears difficult to find the right translation. The task provides the term "heat transport". This yields no useful results in Dipe-D; instead "heat transfer" should be used in the specifications. The test persons often do not find this translation. Dipe-D offers little help in the translation, since the example knowledge representation does not relate them[80].

The second cause is that more than one specification is needed, i.e., that the expressive power of the specification language is insufficient. Only the first test person identifies all three relevant concepts. However, he also identifies an irrelevant concept (see above, under precision). Apparently, he uses an inappropriate specification that yields the relevant concepts as well as an irrelevant concept. This specification gives only a type and a single feature, i.e., not all the features of the task. The test person thus should have either (1) checked each concept in the

80 It may be a solution to extend the knowledge representation by the thought that heat transfer is a subtype of heat transport.

solution for its having the other feature as well, or (2) added another feature to the specification. The latter action would not have yielded all the relevant concepts.

In this task, even when a specification contains the features expressed in a way that coincides with the way they are expressed in the example representation, the specification does not yield all the relevant concepts. In this task, it seems (from the results of the experiments) that at least two specifications are needed for identifying all the relevant concepts.

When two or more specifications are needed, the results of at least one specification need to be recorded or remembered, in order to be able to take them into account. The six test persons do not do this. Instead, they use just one specification, thereby identifying only a subset of the set of relevant concepts.

A possible explanation for this behaviour is that they expect that the tasks are such that a single specification would yield the solutions proper. This is plausible, in the light of their answering the questionnaire that they believe they have identified all the relevant concepts and that the specifications of Dipe-D offer sufficient expressive power. Apparently, both opinions are false. Another possible explanation for the absence of multiple specifications by the test persons is that it puts too high a load on their brains; they may not be able to remember what they have done already and simultaneously think about the steps still to be taken. The real reason was not asked for, and thus is only guessed at. But it is almost certain that the problem would not exist if the specification language had a higher expressive power, in the sense that combinations of specifications are needed. For example, it would be useful if a list of relation types could be given in a specification, instead of a single relation type. The specification then could be solved for each relation type (the relation types thus are treated in a disjunctive manner); at present, the user has to make as many specifications as there are alternative relation types, in order to get the same result.[81] As a second example, the same idea could be applied to the terms (or concepts) in a subexpression: instead, a list of terms could be given. Subsequently, the subexpression could be solved for each of the terms (thus treating them as disjunctive).

In summary, the concept recall is below 100%. We have identified two reasons for this:
– the translation of types and features of the relevant concepts as specified in the tasks into the way they are expressed in the example representation is too difficult,
– the specification language has insufficient expressive power.

8.4.2 Conclusions

On the basis of the above results and discussion, we draw three conclusions. The concept precision in the experiments and the observed behaviour of the test persons lead us to the following conclusion on the first hypothesis:

81 We note that the above-mentioned result of task 1 for test person 1 could also have been improved by using several specifications. In the specification used, the relation type "… developed to do function …" was used. The relation type "… were applied for …" would have yielded the more specific relevant concepts (open and closed Brayton cycle).

Conclusion 1: The precision hypothesis cannot be rejected on the basis of these experiments.

Thus, using Dipe-D it is possible to identify concepts of an information need without identifying irrelevant concepts.

On the basis of the values for concept recall, we draw our second conclusion.

Conclusion 2: The recall hypothesis is to be rejected.

On the basis of the log files and our discussion, we draw a third conclusion:

Conclusion 3: The main causes for the low concept recall in the experiments are: (1) difficulties in translating the expression of types and features in the tasks into the types and features in the example knowledge representation, and (2) insufficient expressive power of the specification language of Dipe-D.

8.4.3 Future research

For future research, our suggestions regard (1) experiments with different types of test persons, (2) the reduction of the translation problem, (3) the extension of the expressive power, and (4) the simplification of the explanation function.

Experiments with different types of test persons

The test persons in the concept-identification experiment are selected: each person has a Master's degree in engineering of a technical university (the degrees are received less than 10 years before the experiment), or is in his final year for such a degree. It is generally accepted that technically educated persons, at university level, have well-developed analytical skills and often the ability to think in a structured way. It may be that these skills combine well with the Dipe-D specification language, since this language is rather structured too. Persons having a different kind of university degree, or a less elaborate education, may not be able to work as effectively with Dipe-D or may need additional training. It would be interesting to test whether such a difference occurs.

The reduction of the translation problem

The first cause of Dipe-D's low concept recall is the failure of the test persons to translate given features and types adequately into those present in the example representation consulted by Dipe-D: the translation problem. For reducing this problem, we suggest to offer more suggestions to Dipe-D's and Dipe-R's users. By having more suggestions, they may become better able to do the translation. One way of offering more suggestions is to introduce a new relation type especially for this purpose. A relation of this type, to be called "… is conceptual part of …" should represent the relationship between a concept and its constituent concepts. For example, in the concept heat transfer we recognise as constituent concepts heat and transfer. Heat and transfer each can be related to heat transfer, by using (twice) the "… is conceptual part of" relationship. This increases the number of connections among closely related concepts, and as such may well increase the number of suggestions for translating.

Of course, not all closely related concepts are related by their being conceptual parts. Therefore, for being able to relate such concepts in a more general sense, a relation type "… is closely related to …" may be introduced. However, this may lead not only to more useful suggestions but also to many useless suggestions, since the type of relationship may not be the required type. The ratio of useful suggestions versus useless suggestions can be determined

experimentally. A third manner of extending the knowledge representation is to represent a domain into more detail. For example, heat transfer and heat transport were not related, but heat transfer can be regarded a subtype of heat transport. This could be represented.

The extension of the expressive power

The second cause of low concept recall is a lack of expressive power of the specification language. The required expressive power is to combine several specifications of the present specification language. We make three suggestions for how this may be realised:

a. The hierarchical classification of relation types

Concepts are hierarchically related. The specification language utilises this in its subexpression type "… any type of <term>", and in other subexpression types that include a part with "any type of <term>". In the specification language, relation types can only be specified individually. A possible extension could be to introduce hierarchies of relation types as well in Dipe-R. This is already possible in the present version of Dipe-R. The specification language then should be extended in a manner similar to that for concepts, e.g., the subexpression type # would obtain a variant: "… related by any type of relation type <r> to <term>".

This was foreseen in the development of Dipe-D, but not introduced so far because we expected that the specification language would become too complex. This reason still holds; an extension as described thus should be combined with an effort to present the various subexpression types of Dipe-D in a very transparent way (see Subsection 8.10.1).

b. The collection of the results of several specifications

The results of a specification, i.e., a collection of concepts (and possibly the specification itself) should be storable and retrievable, such that the results of several specifications can be collected and subsequently a query can be formulated using all the concepts.

c. The introduction of a disjunction operator

We mentioned this solution above. In the specification language, each slot in a subexpression allows for a single relation type or a single term. Thus, if the user wants to try two different relation types or terms, he has to make two specifications and thus also to remember the results of one or more previous specifications. In a new version of Dipe-D, the disadvantage could be overcome by introducing a disjunction operator, and thus to allow for the disjunction of several specifications of the present specification language. Alternatively, or additionally, it could be made possible to fill each slot with more than one relation type or term. The presence of two or more such variables should be interpreted as a disjunction. For example, "I want the concepts that are related to …" with "heat transfer / heat transport" should be solved twice, and the resulting two sets of concepts should be unified.

Simplification of the explanation function

Simplifying the explanation function can increase the efficiency of a search session. As already mentioned, the explanation function was often used only to determine a relation type; this happened when the relation type was not given in the specification. The test persons then were overwhelmed with information, from which they only used the relation type. For increasing the

efficiency, Dipe-D's explanation function may be altered such that it shows the relation type information clearly distinguished among the rest of the explanation (e.g., by highlighting it).

8.5 Design of experiment for the transformation step

The section consists of eight subsections, presenting the design of the experiments for the second step of query formulation. Subsection 8.5.1 states the goal and hypothesis of the experiment. Subsection 8.5.2 outlines the experiment and the Subsections 8.5.2 - 8.5.6 provide detailed features. Subsection 8.5.7 describes what information will be collected in the experiments. Subsection 8.5.8 tells how a conclusion is drawn from the information collected.

8.5.1 Hypothesis

The experiment serves to test whether the second step of Dipe-D produces effective Boolean queries as predicted by the theory of Chapter 6. Since the second step of Dipe-D offers the formulation by computer as an alternative for query formulation by hand, we are particularly interested in the effectiveness of both ways.

The techniques of Dipe-D's transformation step are in essence a formalisation of search behaviour observed in human beings (see Chapter 4). Therefore, if the knowledge representation contains the concepts to be used in the queries, and their various designations, the query-formulation techniques may well yield effectiveness in the same order of magnitude as manual query formulation does. We do not expect that Dipe-D outperforms persons exceptionally skilled in query formulation, since such persons may include subtleties in the queries that are not captured in our query-formulation techniques. We base our third hypothesis on these expectations for Dipe-D's effectiveness.

Hypothesis 3

The queries created by the second step of Dipe-D are on average as effective as those created manually by test persons familiar with Boolean document-search processes.

8.5.2 Outline of the experiment

The experiment uses the example knowledge representation. As a result, it provides only an impression of the effectiveness of Dipe-D's second step, and no solid conclusions. Moreover, the precision enhancement technique of Dipe-D is not tested, since the example knowledge representation does not contain the information necessary for that purpose (i.e., homonymous terms, and the subject areas of the different concepts of such terms).

The experiment comprises three transformation tasks. In each task, a collection of represented concepts was given, and Boolean queries were created, by Dipe-D, and by each member of a group of test persons. The concepts were given to Dipe-D by their identifiers as present in the example representation, and to the test persons by (identifying) designations. The concepts have as subject area refrigeration. Next, the queries were matched in a Boolean retrieval system. The test persons were not present during the matching process. Hence, the document-search process did not contain relevance feedback. Once the matching process was finished, the documents matched were judged for relevance (by their content). Two relevance judges, both engineers who have Master's degrees and working experience in refrigeration, made the judge-

ments. On the basis of the relevance judgements, the precision and the recall of the queries were calculated, as well as the effectiveness; the latter as a function of the precision and the recall. Finally, the effectiveness of the queries created by Dipe-D was compared to the effectiveness of the queries created by the test persons.

The details of the experiment are provided below. Subsection 8.5.3 provides information on the document collection used, Subsection 8.5.4 describes the equipment used in the experiment, Subsection 8.5.5 describes the search tasks, and Subsection 8.5.6 the test persons.

8.5.3 The document collection

The document collection consists of patent documents: the collection comprises the abstracts of all US patents issued since 1972. This includes documents in the subject area of our sample representation. In the experiment, we used a part of the patent document only: the abstract. We have two reasons for choosing abstracts.

The first reason for using abstracts only is to reduce the noise of the experiment. The abstract of a patent document describes usually the subject of the invention well. The whole patent document does this too, but also discusses other (though related) subjects. The latter subjects become noise in a full-text indexing process, in the sense that they are indexed as well as the real subject of the document. They thus lead to a reduction of the average precision of a document-search process using these indexes. With less noise, fewer test persons and tasks are required to measure the effectiveness of transformation by Dipe-D and by the test persons.

The second reason for using abstracts only is that the evaluation of an abstract takes less time than the evaluation of the entire publication, while the quality of the evaluation is not essentially reduced. Since the abstract contains a detailed characterisation of the invention in the document, the relevance can be judged by the abstract almost as good as by the entire document.

8.5.4 Equipment

The queries created by our implementation of Dipe-D (see Chapter 7), and those created by the test persons, are processed on a Personal Computer having a Pentium processor, and running the Windows95 operating system.

The queries are matched by a retrieval system on the WWW and accessed via Netscape Navigator 3.01. The (commercial) database provider Questel offers the retrieval system as a free service[82]. It offers a matching process with a Boolean query language (see Subsection 2.1.3); the query language offers the conjunction, disjunction, and negation operators, as well as proximity operators, parenthesising, and phrase matching. Its relevance ranking is not used. Their patent numbers and titles list the documents matched. Each pair of number and title is shown on a separate line. Each of these lines is hyperlinked to the front page of the document, i.e., the patent publication (see Subsection 8.5.2). The document comprises an abstract, and miscellaneous descriptors not related to the content of the patent, e.g., the name of the inventor.

82 URL: http://www.qpat.com.

8.5.5 The search tasks and relevance criteria

In the experiment, there are three query formulation tasks to be completed by each test person. In each task, a test person is asked to formulate a query by hand, with given concepts. The concepts are provided in natural language, along with quasi-synonyms and synonyms. The test persons are asked to aim at a reasonably high precision (as opposed to a high recall). We explained them that the reason for this is that in this manner we expect fewer documents matched than when a high recall is asked for; as such, the number of documents to be evaluated is reduced, and thereby the amount of effort needed for the evaluation.

The documents matched for each query are, as we already mentioned, evaluated by two relevance judges. Both judges have a Master's degree in mechanical engineering, with refrigeration as major subject, and working experience in the field of the refrigeration area. The two judges have established the relevance criteria for the three tasks. Guided by these criteria, they have reached consensus on the relevance of each document judged. They have judged the relevance of the documents for each task, in two steps. The first step comprised of judgement on a scale of four values (very irrelevant, slightly irrelevant, slightly relevant, very relevant). This yielded four collections of documents, per task. The second step comprised of the merging of the two collections of irrelevant documents, as well as the merging of the two collections of relevant documents, per task. This yielded two collections of documents per task: a collection of irrelevant documents and a collection of relevant documents. On the basis of the latter two collections, we calculated the precision, the recall, and the effectiveness of the document-search process (see below). The three tasks and their relevance criteria are presented in the Appendix E.1.

8.5.6 Test persons

Since we want to compare Dipe-D's query formulation to that of average users, the test persons must have query formulation knowledge resembling that of average users. They have at least some experience using document-search processes with Boolean queries. They are not experts in query formulation, but they are familiar with it. We require that they more or less understand the subjects of the tasks, in order to avoid that a lack of understanding reduces their performance. Therefore, we require that they have a Master's degree in engineering or in exact science.

8.5.7 Information collected

In the experiments, we collect information of the following three types. The reasons are evident.

– The queries, as created by Dipe-D and by the test persons.
– The abstracts of the documents matched.
– The relevance judgements of the documents matched, by the two relevance judges.

8.5.8 Evaluation of the results

We first present the procedure of the evaluation, and then provide the definitions for the performance indicators used in the experiment.

Evaluation approach

The approach is straightforward. First, each query is matched. Then, the documents matched are judged for relevance; the criteria are stated in the Appendix E. Next, for each test person, we

calculate the precision, the recall, and the effectiveness. Then, the averages of the precision, the recall, and the effectiveness are compared to the precision, the recall, and the effectiveness obtained by each of the three recall variants of Dipe-D (basic, weak recall enhancement, strong recall enhancement). On the basis of this comparison, the hypothesis is evaluated. The conclusion is indicative only, since the number of experiments is too small for obtaining significant differences. Large numbers are required for statistical significance, since the precision and the recall of queries matched typically have a high variance.

Definitions

The precision is defined according to Subsection 1.1.2. The recall is defined differently to the definition of that section: instead of the total number of relevant documents in the document collection, we use relative recall, i.e., we use the total number of relevant documents *identified in the experiment, by all test persons.* This relieves us from the laborious task of evaluating *all* documents in the patent document base consulted. The use of relative recall implies that the recall can be used for our purpose, i.e., for comparing the performance of Dipe-D to that of the test persons, but that it may not be interpreted as an absolute performance indicator. As such, the recall cannot be compared to the recall measured in other experiments.

The effectiveness is a function of the precision and the recall. The effectiveness simplifies a comparison of the quality of queries, since it is a single number. However, a difficulty is to choose an appropriate function.

Our definition of effectiveness is derived from the definition of E-measure by Van Rijsbergen (1979): we use the reciprocal value of the E-measure.

$$Eff = \frac{1}{\frac{\alpha}{P} + \frac{1 - \alpha}{P}}, 0 \leq \alpha, P, R \leq 1$$

Here, the symbols refer to:

Eff	the effectiveness,
α	a constant that determines the importance of recall relative to precision,
P	the precision,
R	the recall.

We choose $\alpha = 0.5$, thus giving equal importance to both precision and recall. We note that if $\alpha = 0$, then Eff = R, and that if $\alpha = 1$, then Eff = P. Thus, recall and precision can be considered special cases of the above effectiveness measure.

8.6 Results

The results of the queries, per test person and task, are presented in the Appendix E.2. Some of the queries by the test persons were repaired, in the sense that small mistakes were repaired. The reason for this is that without the repairs thousands of documents would be matched and thus would need to be evaluated. That would have been infeasible. The result of the corrections

is that the measured performances of the test persons are better than their real performances. As a result, the comparison of Dipe-D to the test persons is more demanding for Dipe-D.

The average results of the query tasks are shown below. Table 8.6 shows the search results per person, and for Dipe-D. Table 8.7 shows the search results per task, averaged over the test persons. Table 8.8 shows the search results, averaged over all persons and tasks.

We note that test person 8 scored a very low recall. When asked, he declared that he had focused entirely on a very high precision, thereby ignoring the negative effect on the recall. Since the effectiveness quickly drops to zero if either the precision or the recall approaches zero, the effectiveness of this test person's queries is also very low.

Person (#)	Average precision (%)	Average recall (%)	Average effectiveness (%)
1	45.21	57.45	50.6
2	30.42	40.30	34.7
3	50.72	61.84	55.7
4	63.91	30.64	41.4
5	54.67	29.36	38.2
6	43.95	56.74	49.5
7	25.71	80.00	38.9
8	66.67	4.04	7.6
Dipe-D	45.17	41.84	43.4
Dipe-D wre[83]	35.19	41.84	38.2
Dipe-D sre[84]	47.69	61.84	53.9

TABLE 8.6: *Average search results per test person, and for Dipe-D.*

Task	Precision (%)	Recall (%)	Effectiveness (%)
1	8.33	17.5	14.1
2	73.21	68.75	70.9
3	61.43	48.94	54.5

Table 8.7: Average search results per task, averaged over the test persons.

Precision (%)	Recall (%)	Effectiveness (%)
47.61	45.0	46.3

TABLE 8.8: *Search results, averaged over all test persons and tasks.*

83 wre: weak recall-enhancement technique used.

84 sre: strong recall-enhancement technique used.

8.7 Evaluation

The evaluation is divided into three subsections. In Subsection 8.7.1, we discuss the results in two parts: (1) differences in effectiveness among the three query tasks, and (2) differences in effectiveness among the three variants of Dipe-D. In Subsection 8.7.2, we discuss other interesting lessons learned from the results. In Subsection 8.7.3, we make suggestions for future research.

8.7.1 Discussion

The results show two types of differences in effectiveness: (1) differences in effectiveness among the three query tasks, and (2) differences in effectiveness among the three variants of Dipe-D.

Differences in effectiveness among the three query tasks

Dipe-D's average overall precision, recall, and effectiveness over all three tasks (43.4%, see Table 8.6) are almost as high as the averages over all test persons (46.3%, see Table 8.8). Moreover, Dipe-D shows similar performance differences over the three tasks as the test persons do; see Table 8.7 and the Appendix E.1. In other words, both Dipe-D and the test persons performed worst in task 1 and best in task 3, with task 2 in the middle. The finding that the performance is similar in all tasks could be expected, since the query formulation techniques operate similar to human beings; they are formalisations of a pattern observed in human query formulation (see Chapters 3 and 4).

The finding that there is a large difference in effectiveness among the three tasks (both for Dipe-D and the test persons) can be explained by the retrieval system used: full-text indexing, and Boolean matching. In this type of retrieval system, the effectiveness depends on the specific words and terms used in the content of the documents.

In task 1, the documents contain the query terms, but deal with entirely different subjects. In task 2, most documents that contain the terms "heat pipe" and "bellows" apparently deal with "bellows heat pipe", and thus are relevant. Task 3 requires only 1 or 2 words in the query, and these words are in the context of the document collection unambiguous. Thus, a document matched probably deals with the relevant subject.

Differences in effectiveness among the three variants of Dipe-D

The three variants of Dipe-D show differences in effectiveness (see Table 8.5) that do not coincide entirely with the differences expected on the basis of the theory underlying these variants (see Chapter 6).

From the theory, we expect that, on average, the basic query-formulation technique has the lowest recall; it creates queries by literally using the designations of the concepts about which documents are needed, and their synonyms. In contrast to the theory, the weak recall enhancement technique should, on average, have a higher recall, since it connects the words in the phrases by adjacency operators, and leaves out the articles and prepositions. This may of course diminish the precision. The strong recall enhancement technique is as the weak recall enhancement technique, augmented by including quasi-synonyms of the designations of the concepts about which documents are needed. This should, on average, further increase the recall, but may further reduce precision, since quasi-synonyms may also have other meanings than those intended.

The unexpected pattern observed in the experiments is that in task 1 both the basic

technique and the weak recall enhancement technique yield no relevant documents. In that task, the strong recall enhancement technique yields a precision of 37.5%, and a recall of 60%. Apparently, the weak recall enhancement technique does not have effect; this is not according to the average expectation of the theory, but may happen incidentally according to our theory (since it is a heuristic relationship only). The strong recall enhancement technique does have effect; this is in line with our theory.

In tasks 2 and 3, the recall was the same for all three variants (i.e., the number of relevant documents matched was the same in all three variants). This is possible according to our theory, but not to be expected on average. A possible explanation for this deviance is that apparently there are no additional relevant documents present in the document collection. If the latter assumption holds, recall enhancement does not yield more relevant documents than the basic technique does.

In tasks 2 and 3, the average precision drops between the basic variant and the weak recall enhancement technique; this is in line with the theory. The weak and strong recall enhancement techniques yield identical precision values. Although not in line with the average expectation according to the theory, for this case it is to be expected: the required quasi-synonyms are not present in the sample knowledge representation, and hence the weak and strong recall enhancement techniques yield identical queries.

8.7.2 Conclusion

From the Table 8.8, we learn that the average effectiveness of Dipe-D's basic query formulation technique in the three tasks of the experiments is a fraction below the average performance of the eight test persons: 43.4% versus 46.3%. Dipe-D's weak recall enhancement technique has a somewhat lower average effectiveness (38.2%) than its basic technique, resulting from an equal recall and a lower precision. Dipe-D's strong recall enhancement technique has a higher average effectiveness than the test persons have: 53.9%.

Even when looking at the effectiveness of the best test persons, Dipe-D performs well: Dipe-D's strong recall enhancement technique has almost as high an effectiveness as the best test person in the experiment: 53.9% versus 55.7%, resulting from an equal recall, and a fractionally lower precision of Dipe-D than the precision of the best test person.

On the basis of these results, it is not justifiable to reject the second hypothesis. However, we realise that this conclusion is rather provisional, since the number of experiments is only small, and thus no statistical significance is obtained.

Conclusion 4: the hypothesis 3 cannot be rejected.

8.7.3 Future research

Obviously, additional tests are needed before a more definitive evaluation can be made. Such experiments should include a larger number of test persons, and a larger number of tasks than in the present experiment. Additionally, the enhancement techniques should be tested more extensively. The strong recall enhancement technique is only tested in task 3, since in the other tasks there are no quasi-synonyms that may reveal the difference between weak recall enhancement and strong recall enhancement. In the experiment, the precision enhancement technique is not tested. Tests of the precision enhancement technique require a knowledge

representation having more than one subject area, since it presupposes the presence of homonymous terms.

8.8 A comparison of Dipe-D to ANS

In Section 6.2, we described ANS as an alternative means for concept identification. In that section, we compared Dipe-D and ANS, on the basis of theory. In this section, we extend the comparison, on the basis of experiments.

Wiesman (1998) did some experiments on ANS, but they provided only the effectiveness of document identification, and not of concept identification. Document identification follows concept identification in a document-search process (documents are identified after query formulation and subsequent matching, with the query containing the designations of concepts). The relationship between the concepts identified on one hand and the documents matched on the other hand is stochastic, and not known. We therefore do not know how well test persons using ANS performed the concept identification.

We will make two comparisons. The first (Subsection 8.8.1) is on ANS and Dipe-D with respect to the circumstances of the concept-identification experiment described above. The second (Subsection 8.8.2) is on an extrapolation of both mechanisms of the first comparison.

8.8.1 The comparison

The comparison is based on the performance of one test person using ANS and our example representation. The latter is converted to the format required by ANS[85]. We created two versions: one without derived thoughts, and one with derived thoughts. In the version without derived thoughts, the user has to make derivations by himself, by combining two or more thoughts presented on the video display screen.

The test person finds two concepts in tasks 1, both relevant; three concepts in task 2, all relevant, and one concept in task 3, relevant. This is shown in Table 8.9.

	Concept recall (%)	Concept precision (%)
Task 1	40	100
Task 2	100	100
Task 3	33.3	100

TABLE 8.9: *The concept recall and concept precision of the additional experiment.*

These answers are found in the representation without derived thoughts, and after activation of the link type filter that removes the relations of the type "is somehow related to". This is the relation type introduced for producing noise (see Section 7.2).

The version with derived thoughts contains many derived subtype relations, obtained by

85 The ANS-format is somewhat poorer: it does not offer a full sentence, but two terms and a relation type.

the transitivity rule. These only clutter the screen; every user will make these derivations by himself. Some other derivations seem informative, however.

The precision is as good as in Dipe-D: only relevant concepts were identified. The two causes for a poor recall in Dipe-D's concept-identification step are likely to be present in ANS as well.

First, a 'translation' is needed in ANS as much as it is in Dipe-D. If no relations among closely related concepts are represented, the user has to do the 'translation' himself. If such relations are represented, both ANS and Dipe-D use them. ANS shows them, and in Dipe-D the specification language uses them, or the Quick Browser shows them.

Second, on the basis of the theory, Dipe-D should outperform ANS (regarding effectiveness) in those four types of cases mentioned in Section 6.2. On the basis of our experiments, we cannot prove such a difference.

We also do not know the differences in efficiency between both tools. It would be interesting to find out how fast the user searches when using the specification language of Dipe-D, and how fast the user searches by using the commands for expanding, diminishing, and filtering as available in ANS.

8.8.2 The comparison extrapolated

In the experiments for the first step, and the above-mentioned additional experiment, the variables have the following values.

– Knowledge representation: small, average density, hardly or no inheritance (features defined at a very specific level).
– Search task: 2-4 features, not entirely at the appropriate abstraction level. The terms used differ somewhat from those in the representation. Each task has several solutions, and
– Test persons: university level, technical/exact.

Below, we extrapolate the results, in a speculative elaboration of how the differences change when the variables are varied. We distinguish three types of variables: (1) those regarding the knowledge representation, (2) the search tasks, and (3) the test persons.

Knowledge representation

We discuss (1) a larger knowledge representation, with the same density, (2) a more dense representation, and (3) a representation with more inheritance.

In a larger representation, we expect no other performance than for the example representation; at least, as long as it has an identical density. The reason for this is that the browsing in a local part of the representation seems identical to that in the example representation.

For a representation having a higher density, the chances of loosing one's orientation seem to increase in ANS. One reason is that the paths to be followed become longer, i.e., more links have to be followed; thus more concepts and their accompanying branches are passed, and more irrelevant branches can be pursued. A second reason is that the number of branches per concept becomes larger, thus more irrelevant branches exist, in which one may loose one's orientation (assuming that not all roads end up in Rome – i.e., that these branches do not more often lead to the proper answer). We mentioned these two reasons already in Section 6.2. Dipe-D will not suffer from such deterioration, since the concepts are identified on the basis of a description by features, regardless the density. Hence, Dipe-D will become more effective than ANS.

For a representation having more inheritance, ANS is also likely to suffer from deterioration. If in ANS the inherited represented thoughts are shown, they lead to a denser network. As mentioned above, a higher density leads to more branches and may result in loosing one's orientation more easily. The result is a lower concept recall.

Search tasks

In a search task, features may be given at an improper abstraction level. This occurs in the experiments above, e.g., "compression that occurs isobaric" was asked for, instead of "isobaric compression of gas". The test persons of Dipe-D adequately dealt with this: they used the subexpression type "any type of".

For Dipe-D, we expect no problems from inappropriate abstraction levels; not even if the level given differs much from the level needed. In ANS, an inappropriate abstraction level requires the user to browse along a path with a length corresponding to the difference in abstraction level of the given and needed concepts. This browsing results in the effectiveness-reducing risks described above.

Another problem may occur if a search task has several concepts as solution. In the experiments above, this already played a role. In Dipe-D, a specification may describe several concepts, and they will all be identified by its solving procedure. In ANS, each concept has to be identified by means of browsing the path from the given features to the concept. Meanwhile, the user needs to keep in mind which paths were already pursued and which not. This may lead to mistakes. We expect that this effect will increase with the number of concepts in the solution. Of course, more research is needed here.

8.9 Lessons learned about Dipe-R

The experiments in Sections 8.2 – 8.6 are tests meant for Dipe-D. In some respects we have tested a system of three entities: (1) Dipe-D, (2) the example representation, and (3) the test persons. Of course, we cannot blame the test persons; they constitute an independent variable. But we can only hold both Dipe-D and the example representation responsible for the outcomes of our tests; we did so in Section 8.4. Below, we extend the discussion, focusing on what lessons we learn about the elements of Dipe-R.

Thought types

The two types of represented thoughts offered by Dipe-R seem well understood, with two exceptions. We base this finding on the concepts identified, on the answers to the questionnaire, and on the comments made during the sessions: all test persons answered affirmative on the question whether they could understand the represented thought types. Two problems with thought types were observed, and these concern the relation types therein. The first problem relates to the pre-represented transitivity property, the second problem relates to two relation types used in the sample representation. Below, we deal with both of them.

The first problem is that the representation of the transitivity property of relation types was not clear to the users. This property was expressed in the second represented thought type (tt_2) of Dipe-R (see Section 5.5). A possible explanation is that this represented thought type is an unnatural way of stating transitivity. This explanation is supported by our finding that several

test persons asked for an eludicidation when they encountered a sentence expressing the transitivity of a relation type during the experiment. Since such an elucidation was given when asked, we believe that the unclarity of the representation of the transitivity property has not influenced the performance of the search process. A more natural way of stating transitivity might use the first sentence type of Dipe-R: "… has_property transitivity".

The second problem is the confusion of relation types. The example representation has several relation types. These are all at the same level, and meant to have no overlap in meaning. At least two test persons understood the relation types "(concept) is_somehow_related_to (concept)" and "(concept) has_feature (concept)" as being more general than the other relation types, such as "(thing) … contains (thing) …." They mistakenly expected that by choosing one of the former two relation types, the other relation types were included.

One test person mentioned that this vagueness in the example representation affects the usability of the specification language for him; we indeed observed him making a mistake in this respect. We note that two other test persons found out by themselves that the relation types were on the same level, and used them accordingly. It thus seems better either to avoid the former two relation types in representations, or explain to the users what each relation type means (e.g., by providing example sentences containing these relation types).

Type information of relations

The type information of relations appears useful. It increases the user's confidence and sometimes also the effectiveness of the concept identification. This became visible in a small experiment, were the same test persons of the concept-identification experiments had to fulfil two tasks using Dipe-D's Quick Browser (see Section 7.1). In this small experiment, Dipe-D consulted a fictitious knowledge representation in medicine (comprising approximately 70 concepts).

In the small experiment, they showed little confidence when relation type information was absent; they stated that they were merely guessing whether the concepts were related in the manner they needed them. When relation type information was present, they became more confident. Moreover, the knowledge representation consulted has several pairs of concepts that have two relations between the same two concepts. As a result, the test persons gave wrong answers in the absence of relation type information, and the right answers in the presence of such information.

Source information

In the experiments, the source information is hardly used by the test persons. This is according to our expectation; almost all thoughts represented have the same source (the author) and there are no conflicting thoughts represented. Thus it is not necessary to assess the relative credibility of thoughts represented. Future experiments may focus on this.

Derivations

The derivation rules are used in each task. However, the predominant type of derivation is that using the transitivity rule applied on subtype relations. There is hardly any inheritance, due to the absence of features to be inherited; the knowledge representation contains many features expressed in the bottom of type hierarchies. These features are not inherited.

The test persons accepted all derivations without inspecting the derivation rules. We base these findings on observations made during the search sessions: most test persons mentioned that the

derivations were obvious, and that they only consulted the explanation facility to learn what represented thoughts were the basis for the derived thoughts. In Chapter 5, we assumed that the derivation rules reflect generally accepted types of inferences. On the basis of the experiments, we thus have no grounds for abandoning that assumption.

8.10 Future research on Dipe-D

On the basis of the experiments and the comparison to ANS, we provide two suggestions for future research on Dipe-D. These have as subjects the integration of ANS en Dipe-D (Subsection 8.10.1), and the ranking of concept identifiers (Subsection 8.10.2).

8.10.1 The integration of ANS and Dipe-D

As discussed in the Subsection 6.2.1, Dipe-D and the graphical explorers each have advantages in specific circumstances. A combination of the two approaches may be effective in more circumstances than each separate approach. The combination comprises an improvement of the presentation by ANS, and an integration of the commands and presentation. We discuss each separately.

Improvement of the presentation by ANS

The presentation by the ANS browser may become better by a modification in the way of dividing the nodes over the screen. In the present ANS browser the nodes are divided more or less evenly around a selected node. In a future version, the subtype hierarchies could be shown with the supertypes shown above their subtypes, and types of identical levels shown at the same height. This is in line with the common way of drawing decision trees and hierarchies by human beings, and provides an improved overview of the type hierarchy.

In addition to this, distinct type hierarchies could be drawn next to each other, showing relations between two distinct type hierarchies essentially as horizontal lines. As such the distinct hierarchies may be recognised more easily (i.e., at a glance), and their connections as well.

The graph shown in figure 8.1 exemplifies the proposed graphical presentation of the domain representation. It shows the designations in the expression

any_concept_that_is any_type_of treatment **that_is_related_to** insanity,

the concepts that meet the expression, and the paths relating the designations and concepts. The background is not shown in the example. Open dots refer to designations, bold dots to concepts. A box contains a concept that meets the expression.

Implementation of a graphical domain presentation as described evokes a number of research questions. Such questions relate to the precise manner of presentation of the type hierarchies, and to the selection of appropriate concepts, relations and designations to be shown as context.

Integration of the commands and presentation

The advantages of Dipe-D over ANS hold for the four types of circumstances mentioned in Subsection 6.2.1. Dipe-D thus seems particularly useful for dense domains and for information needs in which hierarchies and various features of the concepts are known, but not the concepts themselves. Moreover, Dipe-D allows for the specification of collections of concepts, by both

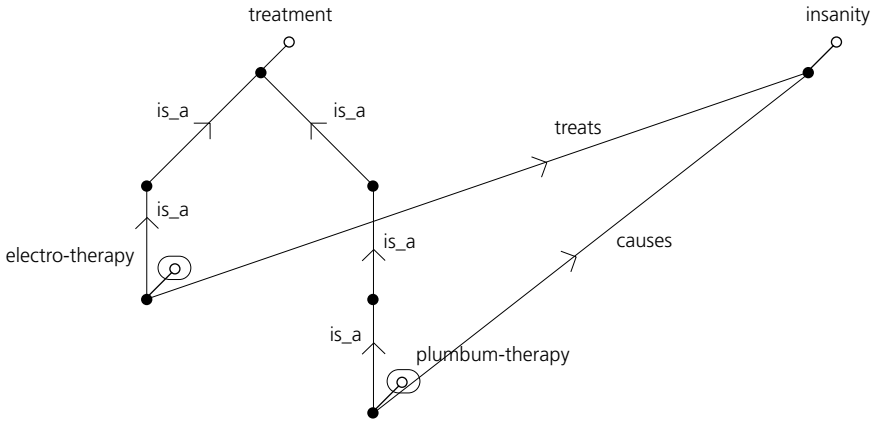

FIGURE 8.1: *Example of different presentation by ANS.*

broad and narrow specifications, while ANS allows for the simultaneous recognition of one (or, incidentally, a few) concepts. Conversely, ANS has the advantage that it can present the domain in a surveyable manner, and that it thus may quickly clarify the relationships in the domain.

An improved domain explorer may be obtained by integrating the browser by Wiesman (1998) with Dipe-D. This must preferably happen after the above-mentioned improvement of the presentation of ANS, and after the improvements on Dipe-D's specification language described in Subsection 8.4.3.

The integration could comprise of showing the solution of a specification in Dipe-D by ANS. As a result, the browsing no longer will take place relation by relation but can be done by entire hierarchies and several (other) relations at a time. Thus, the integrated tool has both the (relatively) good expressive power of Dipe-D and the clear presentation of the ANS browser.

Dipe-D's specification language could be integrated as follows with ANS. Dipe-D's specification language has six main subexpression types. These are presented in Table 8.10, in a compact notation. In this notation, the superscript "d" is equivalent to "any type of", and "c" stands for a given concept (or term), "C" stands for an arbitrary concept (or term), "r" for a given relation type, and "R" for an arbitrary relation type. Behind the compact notation, the number of the corresponding sub-expression type (see Chapter 6) and its description are shown.

Using these six main types the user could specify his information need, to be shown in a seperate box. Subsequently, he could be shown the concepts of the solution in a display window containing ANS. These concepts may be surrounded by related concepts, serving as a context. As such, ANS serves as a presentation means for the solution.

A further, and probably more useful, integration may be realised by offering the specification language also in the ANS display window. The specification language could be given as a menu, having a list with the subexpression types and also an option for the concatenation of two or more subexpressions. The subexpressions replace the simple expand and diminish commands

Abbreviation	Subexpression type	Description
^{d}c	1′	All concepts in the hierarchy that has c as top-concept
R c	4	All concepts having an arbitrary relation to concept c
R ^{d}c	4′	All concepts having an arbitrary relation to a concept in the hierarchy that has c as top-concept
R c	2	All concepts having a given relation of the type r to the concept c
R ^{d}c	2′	All concepts having a given relation of type r to a
		concept in the hierarchy that has c as top-concept
R	6	All concepts having a given relation of type r to an arbitrary concept

TABLE 8.10: *The main subexpression types.*

of ANS for showing all directly related terms of a given term and for undoing the latter. A windows speed menu, showing the subexpression types in the compact notation introduced above could supplement the menu of the specification language. With these measures it becomes possible to specify and evaluate the results of a specification more quickly, and thus it becomes possible to learn the subject area more quickly.

Of course, this idea needs further working out with respect to the precise looks of the user interface (where is what information to be displayed, and where are which commands to be put).

8.10.2 Ranking of concept identifiers in the concept-identification step

As mentioned above, the identification of concepts is similar to the matching of documents. We can extend the analogy, by introducing relevance ranking also in the solving procedure for the identification of concepts. This can be realised by adding a relevance number to each concept of the solution to a specification; the number expresses the degree to which the concept meets the specification, in an interval [0;1]. In the present version of Dipe-D, concepts are identified only if they match all the features specified. In a version that has a ranking, concepts may also have a subset of the features specified. In addition, if a concept has a feature that is closely related to a feature specified (i.e., the two features are closely related in the knowledge representation), it may increase the relevance number of the concepts of which it is a feature.

In addition to the above ranking, the relevance numbers of the concepts may be used in Dipe-D's transformation procedure. The latter procedure may be adapted to produce weighted query keys (see Section 3.3). The advantage of such weights is that it becomes possible to distinguish relatively important query keys from unimportant keys, and thus to improve the relevance ranking of the documents matched (by the retrieval system). The weights can be obtained from the relevance numbers produced in the concept-identification step. A similar idea was presented in Järvelin *et al.* (1996). Additionally, the weights may be adapted by the recall enhancement variant of the transformation procedure: quasi-synonymous designations of a concept may be assigned lower weights than the direct designations. This is readily realised by a multiplication factor that differs for quasi-synonymous designations and direct designations.

Chapter 9
Conclusions

The chapter provides the conclusions on our research. First, Section 9.1 summarises the three research questions investigated in the previous chapters: the approaches, the results, and the conclusions drawn so far. Next, Section 9.2 combines the research into a conclusion on our main research question.

9.1 The research recapitulated

For each of the three research questions investigated in the thesis, we recapitulate the approach, the results, and the conclusions. We note that in each of the investigations the theoretical basis of Chapter 2 is adopted. The three questions are discussed in seperate subsections.

9.1.1 Query formulation

Question 1

The first research question is prompted by the need for more insight into query formulation, as a subprocess of query-based document-search processes. Such insight then should become a starting point for the improvement of query-formulation processes.

How are queries formulated adequately, in a search task that serves the writing of a patent application document?

Approach and results

The first research question is assessed in Chapter 4. The chapter contains a theoretical study of query formulation, as performed in service of another task. The theoretical study is followed by an empirical study of query formulation in patent-search tasks. First, it is described how patent attorneys apply query-based document-search processes. Subsequently, patent-search tasks are simulated ands the query-formulation process in these tasks is studied.

09 Conclusions 03-08-2000 16:36 Pagina 176

Conclusions

The theoretical study of query formulation leads to the conclusion that a distinction should be made between the user's lack of knowledge and the user's understanding of this lack of knowledge (i.e., his information need).

The empirical research leads to three conclusions on patent search, as performed in service of the writing of a patent application.

1. The user's understanding of his knowledge lack (i.e., his information need) changes noticeably, i.e., the development of that understanding plays an essential role in the search process.
2. Concepts are identified by their characteristics (direct or by association).
3. Queries are formulated according to a fixed procedure, in which the change of the information need and the identification methods mentioned above play a role.

Since the above-mentioned fixed procedure is identified by a small-scaled study for the specific task of patent search, the conclusions hold only for the study itself, and any generalisation beyond the study is speculative. Nevertheless, the principles above may be used in any subject area as long as they are used as a prescription for query formulation processes (i.e., not as a description); they do not depend on properties specific to a particular subject area.

9.1.2 Knowledge representation: Dipe-R

Question 2

The existing knowledge representations, such as thesauri, are not ideal for supporting query formulation. Improvements seem possible. These two thoughts prompt our second research question.

> *Is it possible to formulate a representation language for domain knowledge that effectively supports initial query-formulation processes?*

Approach and results

The assessment of our second research question is reported in Chapter 5. The research leads to the development of the knowledge representation language Dipe-R.

Dipe-R meets two requirements that we regard important for effective query formulation. These are not or not entirely met by the existing representations reviewed in Section 5.2. The requirements are:

1. a proper expression of mental states and processes, and
2. the identification of query keys by the hierarchical relations and features of the concepts they designate.

Dipe-R's design meets these requirements by allowing for the representation of two types of thoughts. Each type comprises content information and source information. The content information consists of represented concepts: these serve the representation of quantities, binary relations, symbols, and other concepts. The source information regards person, moment, and origin of the thought. Additionally Dipe-R offers a (simple) derivation procedure and derivation rules for transitivity and inheritance.

09 Conclusions 03-08-2000 16:36 Pagina 177

Conclusion

Our answer to the second research question is divided into a theoretical and an empirical part. The theoretical part is based on the design of Dipe-R (provided in Chapter 5). From the design of Dipe-R, we draw the conclusion that the second research question is answered with an affirmation: it is possible to formulate a representation language for domain knowledge that effectively supports initial query-formulation processes. In theory, Dipe-R is a formulation of such a representation language.

The empirical part of our answer to the second research question is based on the creation of an example representation in Dipe-R (reported in Chapter 7), and based on the experiments with Dipe-D, in which experiments the example representation was used (reported in Chapter 8). In the creation of the example representation, it appeared possible to express much domain knowledge (in the domain of refrigeration). In the experiments with Dipe-D, in which the example representation was used, this represented knowledge has proven to be useful for query formulation. The example representation provided effective support for both the concept identification step and the transformation step. The support being effective is ascribed to Dipe-R's design, in particular the choice of the two represented thought types (each including two concepts and relation type information), the distinction between the content and the expression of represented thoughts, and the choice of the derivation rules.

9.1.3 Query formulation tool: Dipe-D

Question 3

Initial query-formulation processes are often ineffective. We expect that these processes are fit for improvement by techniques that utilise represented knowledge (e.g., expressed in Dipe-R). This expectation prompts our third research question.

> *Is it possible to formulate a knowledge-based technique that effectively supports initial query-formulation processes?*

Approach and results

The research question is assessed in the Chapters 6 to 8. In Chapter 6, the theory is provided, by the development of the query-formulation device Dipe-D. In Chapters 7 and 8 the implementation and experiments are reported. Below, we summarise the theory and the experiments separately.

Theory

The approach for query formulation worked out in Chapter 6 supports, in contrast to the existing approaches, the entire formulation process of an initial query. This is realised by working out the two-step approach suggested in Chapter 4.

1. Concept identification

The concept-identification step serves to develop an expression of the information need by exploring represented knowledge. In an iterative, interactive loop the user identifies the concepts of his information need while developing this information need. The concept-identification loop has the following steps:

09 Conclusions 03-08-2000 16:36 Pagina 178

– specification of concepts in a specification language,
– solving the specification, i.e., identifying in a knowledge representation the concepts that satisfy the specification, and
– presenting the solution, along with an explanation.

By repeating these three substeps, the person is enabled to develop his information need (i.e., his understanding of his knowledge lack). The specification language allows for vague, general descriptions as well as precise descriptions. The person thus is enabled to explore his information need, and to express the information need with an increasing precision.

2. Transformation

The transformation step transforms, automatically, the expression of the collection of concepts identified into a Boolean query. The transformation step has a basic variant, as well as additional techniques for increasing the precision and/or the recall of the queries in a document-search process. In the transformation step Dipe-D again consults a knowledge representation expressed in Dipe-R.

Experiments

Each of Dipe-D's two steps is tested in separation. The concept-identification step is tested by having six test persons each complete three search tasks. In each task, the test person has to identify concepts using Dipe-D (and our example representation in Dipe-R), starting from a written search task. It was verified first that the persons were not able to conceive of the concepts by heart.

On average, the concepts were identified with a concept recall of 70% and a concept precision of 100%. Moreover, the test persons mastered Dipe-D quickly. They used only a small subset of the expressions in the specification language. The concept recall being below 100% seems to be explainable by (1) a translation problem of the search tasks to the specifications in Dipe-D, and (2) insufficient expressive power of the specification language.

The transformation step was tested by comparing transformation by hand to automated transformation by Dipe-D. Eight test persons each completed three query-formulation tasks, in which tasks the terms and their variants were given to the test persons. Dipe-D did the same tasks. Next, the queries were matched by a Boolean search system, having a document collection of patent abstracts.

On average, the test persons and the basic variant of Dipe-D performed equally well (the test persons have an equivalent effectiveness of 46.3%, Dipe-D has 43.4%). The precision enhancement technique was not tested, since the sample knowledge representation does not include appropriate designations for this (homonymous designations, covering more than one subject area). The weak recall enhancement technique performs worse than the basic technique: it yields additional irrelevant documents, but no additional relevant documents. The effectiveness thus drops to 38.2%. The strong recall enhancement technique performs better than the basic technique, hence better than the average test person does.

Conclusions

On the basis of a small set of experiments on Dipe-D, it is currently possible to draw four con-

09 Conclusions 03-08-2000 16:36 Pagina 179

clusions. The conclusions are based on experiments in a specific domain, with test persons who are familiar with the subjects of the domain.

Conclusion 1
Dipe-D supports the user in developing an information need from a given description by features, in natural language, into a collection of concepts satisfying the description; this yields a concept precision of 100%. Thus, Dipe-D enables the user to identify concepts of an information need without identifying irrelevant concepts.

Conclusion 2
In our experiments we did not find that Dipe-D supports the user in developing an information need from a description by features in natural language into a collection of concepts satisfying the description with a concept recall of 100%. However, the concept recall observed was some 70%.

Conclusion 3
The causes for the observed low concept recall are (1) difficulties in translating the expression of types and features in the tasks into the types and features in the knowledge representation, and (2) insufficient expressive power of Dipe-D's specification language.

Conclusion 4
The queries created by the transformation step of Dipe-D are on average as effective as those created manually by test persons familiar with Boolean document-search processes.

For each of the above four conclusions we are aware of the fact that they are only tentative, since the scale of the experiments is small. Statistical significance may be obtained in further experiments.

9.2 Conclusion on the main research question

The overall research question now can be assessed, on the basis of the results and conclusions found in the three research questions. The overall question, as formulated in Chapter 3, is:

How can, in query-based document-search processes, the effectiveness of the formulation of an initial query be improved by utilising represented knowledge?

The results of the first research question provided us with an idea for an improved formulation process for initial queries: a two-step approach in which represented domain knowledge is used.

For the second research question, we investigated the representation of domain knowledge for the purpose of query formulation processes. This has resulted in the design of the knowledge representation language Dipe-R.

The investigations for the third research question have worked out and formalised the aforementioned idea of dividing initial query formulation processes into two steps. The research has resulted in the query formulation tool Dipe-D.

Our experiments with Dipe-D allow for the conclusion that both its steps are effective (see Subsection 9.1.3). Dipe-D's concept-identification step is new and our experiments allow for

09 Conclusions 03-08-2000 16:36 Pagina 180

the tentative conclusion that this step is at least as effective as existing techniques. The transformation step has new elements and our experiments allow for the tentative conclusion that it is as effective as transformation by human beings with some training. Our experiments have also led us to believe that Dipe-R effectively supports Dipe-D.

We did not test the combination of Dipe-D's two steps into an integral process for the formulation of initial queries. However, since the output of the first step is the direct input for the second step and each of the steps is effective, it is reasonable to assume that their combination is effective too. This leads us to the main conclusion of our research:

> *The effectiveness of query-based document-search processes, with full-text indexing and Boolean matching, can be improved by the two-step approach to initial query formulation of Dipe-R and Dipe-D.*

It is obvious that this is a provisional conclusion, as are the underlying conclusions. We believe that this conclusion is sufficiently promising to continue the research towards Dipe-R and Dipe-D. Our present ideas for improving both Dipe-R and Dipe-D were already described in the preceding chapters.

Appendix A
Dipe-R

Pre-represented thoughts

Some thoughts need to be present in all implementations of Dipe-R (see Chapters 5 and 7). These thoughts are presented below. To improve the readability, we present them (where appropriate) as they are expressed by Dipe-R (as opposed to how they are represented in Dipe-R). Moreover, we only present their content (i.e., not their identifier, or source information). Since each pre-represented concept has precisely one designation, there is a one-to-one relationship between a represented thought and its expression.

Concepts used

The existence of a concept is not stated in Dipe-R; by using a concept in Dipe-R, its existence is implied. However, each concept (except the concept 'concept') is introduced as a subtype of concept, by the following thought type: [tt_1, is subtype of, …, concept]. In other words: "… is subtype of concept". The concepts below are used in all implementations of Dipe-R.

Concept
Thought type
Thought type 1
Thought type 2
Relation type
Quantity
Characteristic
Transitivity
Symbol
Word
Stopword
Designation

Relation type concepts:
- has first role
- has second role
- is subtype of
- is allowed for
- has property
- has designation
- has quasi-synonymous designation
- has lexical variant
- contains (in 3 variants)

Quantity concepts:
- All
- Some
- A(n)

An example of a represented thought having a symbol concept:
th(3528, tt_1, has_des#, 27, thing) "(the concept) 27 has designation (the concept) thing".

Represented thoughts for representing relation types

Represented thoughts offering role information for relation types.

For each relation type, we want to represent the first and the second role. But then it needs to be represented first that the two relation types used for this exist, and that they have a first role and second role themselves. This is done as follows (we only mention the content, and abbreviate "thought type 1" to "tt_1", and "thought type 2" to "tt_2"):

[tt_1, is subtype of, has first role, relation type]	"has first role is a subtype of relation type"
[tt_1, has first role, has first role, relation type]	"has first role has first role relation type"
[tt_1, has first role, has second role, concept]	"has first role has second role concept"
[tt_1, is allowed for in, has first role, tt_1]	"has first role is allowed for in thought type 1"
[tt_1, is subtype of, has second role, relation type]	"has second role is a subtype of relation type"
[tt_1, has second role, has first role, relation type]	"has second role has first role relation type"
[tt_1, has second role, has second role, concept]	"has second role has second role concept"
[tt_1, is allowed for in, has second role, tt_1]	"has second role is allowed for in thought type 1"
[tt_1, is subtype of, is allowed for in, thought type]	"is allowed for in is subtype of thought type"
[tt_1, has first role, is allowed for in, relation type]	"is allowed for in has first role relation type"
[tt_1, has second role, is allowed for in, thought type]	"is allowed for in has second role thought type"
[tt_1, is allowed for in, is allowed for in, tt_1]	"is allowed for in is allowed for in thought type 1"

Represented thougths about the relation types subtype and characteristic:

[tt_1, is subtype of, is subtype of, relation type]	"is subtype of is subtype of relation type"
[tt_1, has first role, is subtype of, concept)	"is subtype of has first role concept"
[tt_1, has second role, is subtype of, concept)	"is subtype of has second role concept"
[tt_1, is allowed for in, is subtype of, tt_1]	"is subtype of is allowed for in thought type 1"
[tt_1, is subtype of, have characteristic, relation type]	"have characteristic is subtype of relation type"
[tt_1, has first role, have characteristic, concept)	"have characteristic has first role concept"
[tt_1, has second role, have characteristic, concept)	"have characteristic has second role concept"
[tt_1, is allowed for in, have characteristic, tt_2[86]]	"have characteristic is allowed for in thought type 2"

Represented thoughts for the transitivity property of relations:

[tt_2, has characteristic, all, is subtype of , transitivity]	"all instances of is subtype of have characteristic an instance of transitivity".

Represented thougths about the relation type for expression, "has designation":

[tt_1, is subtype of, has designation, relation type]	"has designation is subtype of relation type"
[tt_1, has first role, has designation, concept]	"has designation has first role concept"
[tt_1, has second role, has designation, designation]	"has designation has second role designation"
[tt_1, is allowed for in, has designation, tt_1]	"has designation is allowed for in thought type 1"

Represented thougths about the relation types for finding query keys, "has quasi synonymous designation", "has lexical variant", "contains":

[tt_1, is subtype of, has quasi synonymous designation, relation type]	"has quasi synonymous designation is subtype of relation type"
[tt_1, has first role, has quasi synonymous designation, concept]	"has quasi synonymous designation has first role concept"
[tt_1, has second role, has quasi synonymous designation, designation]	"has quasi synonymous designation has second role designation"
[tt_1, is allowed for in, has quasi synonymous designation, tt_1]	"has quasi synonymous designation is allowed for in thought type 1"
[tt_1, is subtype of, has lexical variant, relation type]	"has lexical variant is subtype of relation type"
[tt_1, has first role, has lexical variant, concept]	"has lexical variant has first role concept"
[tt_1, has second role, has lexical variant, designation]	"has lexical variant has second role designation"
[tt_1, is allowed for in, has lexical variant, tt_1]	"has lexical variant is allowed for in thought type 1"

86 This choice is arbitrary; it could have been tt_1 as well. With tt_2 it is somewhat more expressive.

[tt_1, is subtype of, contains[87], relation type]	"contains is subtype of relation type"
[tt_1, has first role, contains, designation]	"contains has first role designation"
[tt_1, has second role, contains, word]	"contains has second role word"
[tt_1, is allowed for in, contains, tt_1]	"contains is allowed for in thought type 1"

Represented thoughts for organising symbol concepts:

[tt_1, is subtype of, stopword, word]	"stopword is subtype of word"
[tt_1, is subtype of, designation, symbol]	"designation is subtype of symbol"
[tt_1, is subtype of, word, symbol]	"word is subtype of symbol"

87 This type of contains is not transitive. In many represented domains, another relation type named contains will exist, and may be transitive. That type will have a different identifier, thus can be distinguished.

Appendix B
Dipe-D's expression types

The expression types of Dipe-D's specification language (see Chapter 6) are summarised below. In the summary of expression types, the optional use of "not" is indicated between angle brackets. The general rule is that the negation is used preceding a sub-expression.

<information need> ::= <expr>
<expr> ::= <e1> | <e2> | <e3>
<e1> ::= **I_want_the_concepts_that** <full_subexpr>
<e2> ::= <expr> **and_the_concepts_that** <full_subexpr>
<e3> ::= <expr> **and_that** <full_subexpr>

<full_subexpr> ::= <subexpr> | <**not** subexpr>
<subexpr> ::= <s1> | <s1'> | <s2> | <s2a> | <s2b> | <s2'> | <s2'a> | <s2'b> | <s3> | <s3'> | <s4> | <s4'> | <s5> | <s6> | <s6a> | <s6b> | <s7> | <s7'>

<s1> ::= **have_des** <designation>
<s1'> ::= **are_any_type_of** <designation>

<s2> ::= **have_a_relation_of_type** <relationtypename> **to** <designation>
<s2a> ::= **are_the_source_of_a_relation_of_type** <relationtypename> **to** <designation>
<s2b> ::= **are_the_destination_of_a_relation_of_type** <relationtypename> **to** <designation>
<s2'> ::= **have_a_relation_of_type** <relationtypename> **to_any_type_of** <designation>
<s2'a> ::= **are_the_source_of_a_relation_of_type** <relationtypename> **to_any_type_of** <designation>
<s2'b> ::= **are_the_destination_of_a_relation_of_type** <relationtypename **to_any_type_of** <designation>

<s3> ::= **have_the_same_type_of_relation_to** <designation1> **as** <designation2> **has**
<s3'> ::= **have_the_same_type_of_relation_to_a_type_of** <designation1> **as** <designation2> **has**

<s4> ::= **are_related_to** <designation>
<s4'> ::= **are_related_to_any_type_of** <designation>

<s5> ::= **are_related_to_the_same_type_of** <designation1> **as** <designation2>

<s6> ::= **have_a_relation_of_type** <relationtypename>
<s6a> ::= **are_the_source_of_a_relation_of_type** <relationtype>
<s6b> ::= **are_the_destination_of_a_ relation_of_type** <relationtype>

<s7> ::= **are_any_type_of** <designation1> **{preposition}** <designation2>
<s7'> ::= **are_any_type_of** <designation1> **{preposition} any_type_of** <designation2>

MP[T, e1(full_subexpr)] = MP[T, full_subexpr]
MP[T, e2(expr, full_subexpr)] = {MP[T, expr], MP[T, full_subexpr]}
MP[T, e3(expr, full_subexpr)] = MP[T, expr] ∩ MP[T, full_subexpr]

MP[T, **not** subexpr] = {concept | T ⊨ concept ∈ T} - MP[T, subexpr]

MP[T, s1(designation)] = [T, {concept | T ⊨ have_des(concept, designation)}]
MP[T, s1'(designation)] = [T, {concept | T ⊨∃concept1: have_des(concept, designation) ∨ (relation(subclass, concept, concept1) ∧ have_des(concept1, designation))}]

MP[T, s2(relationtypename, designation)] = [T, {concept | T ⊨∃concept1: have_des(concept1, designation) ∧ (relation(relationtypename, concept, concept1) ∨ relation(relationtypename, concept1, concept))}]
MP[T, s2a(relationtypename, designation)] = [T, {concept | T ⊨∃concept1: have_des(concept1, designation) ∧ relation(relationtypename, concept, concept1)}]
MP[T, s2b(relationtypename, designation)] = [T, {concept | T ⊨∃concept1: have_des(concept1, designation) ∧ relation(relationtypename, concept1, concept)}]

MP[T, s3(designation1, designation2)] = [T, {concept | T ⊨∃concept1, ∃concept2, ∃relationtypename: have_des(concept1, designation1) ∧ have_des(concept2, designation2) ∧ relation(relationtypename, concept2, concept1) ∧ relation(relationtypename, concept, concept1)}]
MP[T, s3'(designation1, designation2)] = [T, {concept | T ⊨∃concept1, ∃concept2, ∃relationtypename: have_des(concept1, designation1) ∧ have_des(concept2, designation2) ∧ relation(subclass, concept3, concept1) ∧ relation(relationtypename, concept2, concept3) ∧ relation(subclass, concept4, concept1) ∧ relation(relationtypename, concept, concept4)}]

MP[T, s4(designation)] = [T, {concept | T ⊨∃concept1, ∃relationtypename: have_des(concept1, designation) ∧ (relation(relationtypename, concept, concept1) ∨ relation(relationtypename, concept1, concept))}]
MP[T, s4'(designation)] = [T, {concept | T ⊨∃concept1, ∃concept2, ∃relationtypename: (have_des(concept1, designation) ∧ (relation(relationtypename, concept, concept1) ∨ relation(relationtypename, concept1, concept))) ∨ (have_des(concept1, designation) ∧ relation(subclass, concept2, concept1) ∧ (relation(relationtypename, concept, concept2) ∨ relation(relationtypename, concept2, concept))}]

MP[T, s5] = [T, {concept | T ⊨∃concept1, ∃concept2, ∃concept3, ∃relationtypename1, ∃relationtypename2: have_des(concept1, designation1) ∧ relation(subclass, concept3, concept1) ∧ have_des(concept2, designation2) ∧ relation(relationtypename1, concept2, concept3) ∧ relation(relationtypename2, concept, concept3)]

MP[T, s6(relationtypename)] = [T, {concept | T ⊨∃concept1: (relation(relationtypename, concept, concept1) ∨ relation(relationtypename, concept1, concept))}]
MP[T, s6a(relationtypename)] = [T, {concept | T ⊨∃concept1: relation(relationtypename, concept, concept1)}]
MP[T, s6b(relationtypename)] = [T, {concept | T ⊨∃concept1: relation(relationtypename, concept1, concept)}]

Appendix C
Handout for the test persons

Introduction

We are testing a new tool for information retrieval (Dipe-D; Dipe stands for 'Description and Identification (of concepts) by (their) Properties', 'D' stands for 'Device') and a new knowledge representation language for supporting the tool (Dipe-R). This tool and language should reduce the well-known problem of finding precisely the documents relevant to your information need, in a large database of documents. You may have experienced this problem yourself, e.g., when you searched the WWW with a search engine, such as Alta Vista.

In our experiments, we test the effectiveness of Dipe-D. Dipe-D supports the composition of queries in a document-search process. It renewes (and hopefully improves) the query composition process by structuring it into two steps: (1) exploring and expressing an information need, and (2) formulating a Boolean query. Dipe-D consults represented knowledge, expressed in Dipe-R.

We ask you to help us in the experiments by doing two comparative tasks, plus three absolute tasks. In each of the tasks, you need to identify concepts using Dipe-D.

Instructions

– Answer the initial questions below. These questions serve to verify whether you qualify as a test person.
– Read the description of Dipe-D, and the example of a search task.
– Do the two tasks of the auxiliary experiment (using Dipe-D entirely or only partly, according to the instructions given by the test supervisor).
– Do the three tasks of the main experiment (with Dipe-D).
– Answer the task-specific questions.

If you need more information, please ask the test supervisor. For instance, if the search tasks are not clear, do ask for clarification!

Initial questions

Answer these questions before you do the search tasks. They serve to verify that your level of knowledge matches the level needed.

1. Do you have experience with (Boolean) document search processes? If so, what experience?
2. What educational level do you have (e.g., university degree, or: 3rd year student)?
3. In what subject area are you educated (e.g., mechanical engineering), and at what level? (university undergraduate/year, master's degree, …). How long ago did you finish the education?

Description of Dipe-D

Dipe-D creates Boolean queries in a two-step process, in each step consulting a knowledge base (expressed in Dipe-R).

The two steps are:

Step I. The exploration of your information need. In this step, you develop your understanding of what concepts you need documents about. For this purpose, the tool provides an iterative loop of four sub-steps:

1. you describe concepts by their features (or characteristics, properties), using a specification language,
2. the tool identifies the concepts that satisfy the description (thereby consulting the knowledge base),
3. the tool presents the concepts, and an explanation,
4. you select concepts from the solution.

This loop is meant to allow you to develop your information need. In other words, you get to know about which concepts you need information. You may repeat the loop as many times as you want. We suggest that you start by a rough description of your information need, and refine the description in each iteration. For instance, you may start by asking for "all concepts related_to_any_type_of *device*" and end asking for "all concepts that_are_any_type_of *thing* and_that_have_a_relation_of_type *was_developed_for_purpose* to_any_type_of *compression*."

The specification language, mentioned in sub-step a, enables one to describe concepts by their features. It assumes that the domain knowledge can be organised in type hierarchies, and that a feature of a concept can be represented as a relation between that concept and another concept. An example domain is shown in figure C.1.

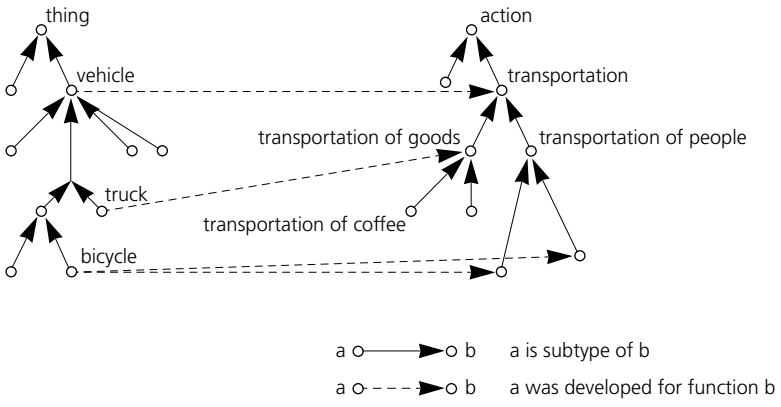

FIGURE C.1: *Example domain knowledge.*

The specification language is exemplified below.

I want the concepts that are any type of *thing*

I want the concepts that are any type of *thing* and that are related to *transportation*

I want the concepts that are any type of thing and that are related to any type of *transportation*

The first expression yields all concepts in the left type hierarchy. The second expression yields "vehicle". The third expression yields "vehicle", "truck", and "bicycle". The details of the specification language will become apparent when you try it in Dipe-D.

Step II. The formulation of a Boolean query. In this step, the tool automatically formulates a Boolean query with the concepts that you selected.

The query then can be matched by a standard Boolean information retrieval system. In the experiments, you will use the first step of Dipe-D only.

Example of a search task

The following is an example of a search task: "You are looking for documents describing processes in which a working fluid undergoes state-changes. The process comprises isobaric addition of heat and isentropic compression and isentropic expansion as sub-processes." Try this task, and ask the test supervisor for instruction.

Tasks for the auxiliary experiment

Rely on the information given by Dipe-D. This may conflict with your own knowledge, since the subject area (medicine) is partly fictitious.

1. Find diagnostic methods for hysteria.
2. Find diseases that can be treated by gamma therapy.

Tasks for the main experiment

1. You are looking for documents describing thermal processes. Each of these processes comprises several steps, forming a closed loop. Each of these processes occurs in gaseous state. Each of the thermal processes can be used for cooling. Each of the thermal processes comprises isobaric compression as a process-step.
2. You are looking for documents describing a regenerative thermal process. The process is cyclic, and has as process-steps: heating of gas at a constant volume, and isothermal expansion.
3. You are looking for documents describing a type of device that serves to transport heat from one point to another. The heat transport can be switched on and off, due to the presence of a bellows.

Task-specific questions

Answer these questions after finishing the tasks. The answers enable us to trace the causes of differences in performance between the old tool and the new tool.

Concerning the tasks you did on Dipe-D (the new tool)

1. Are the expressions of the specification language clear? If no, what is unclear?
   – Do they offer sufficient expressive power to express your information need (as arisen from the search task)? If no, what was it that you could not express?
   – Does the Quick Browser help you in finding concepts?

b. Is the meaning of the statements in the explanation clear? If no, why not?

c. Do you think you have adequately identified the concepts of your information need?

d. How confident are you in your answer to the former question? (very unconfident/ unconfident/ neutral/ confident/ very confident)

If you have any comment on the experiments, please feel free to comment.

Appendix D
The concept-identification experiment

D.1 Solutions of the concept-identification experiment

Task 1	Task 2	Task 3
open Brayton cycle, closed Brayton cycle, Brayton cycle, Ackerett-Keller cycle, and Ericsson cycle.	Stirling cycle, Philips cycle, and Vuilleumier cycle.	bellows type heat pipe flexible heat pipe, and thermal switch heat pipe.

The table shows the solutions of the concept-identification experiment reported in Chapter 8. Each solution is expressed in the example knowledge representation, and can be found using Dipe-D in at least one manner of specification[88].

D.2 Description of the completion of a search task

Below, we provide a description of the completion of a search task. The example is based on the logfile of the communication between the test person and Dipe-D, and on the author's notes, taken during the session. We chose test person 1, task 2. This choice is arbitrary, in the sense that all persons and tasks qualify as examples. However, the log of task 2 are somewhat longer, and therefore we did not choose that task.

The descriptions of the completion of the other search tasks (by all test persons), and the logfiles of these tasks, are provided on WWW: http://www.cs.unimaas.nl/~pol/sessions.htm.

88 For each task, one derivation of the solution is given in http://www.unimaas.nl/~pol/answers.htm

Test person 1, Task 2

The test person starts with a rather precise specification, in the right direction by using the concept regenerative thermal cycle:

"I want the concepts that are any type of regenerative thermal cycle
and that are related to any type of isochoric process."

This yields no concepts, due to a bug in the example knowledge representation: the specific types of isochoric processes are not represented as subtypes of an isochoric process. This bug has been fixed after the test person had finished his tasks. Fortunately this bug does not prevent the test person to find the solution; he chooses a different feature, using knowledge concerning the relation type as obtained in task 1:

"I want the concepts that are any type of regenerative thermal cycle
and that have a relation of type comprise to any type of isothermal expansion of gas."

This yields four concepts: the three concepts of the solution proper, and Ericsson cycle. He verifies the solution in the explanation facility, and then adds another feature:

"I want the concepts that are any type of regenerative thermal cycle
and that have a relation of type comprise to any type of isothermal expansion of gas
and that are related to any type of heating of gas."

This yields the same four concepts. He adds another feature:

"I want the concepts that are any type of regenerative thermal cycle
and that have a relation of type comprise to any type of isothermal expansion of gas
and that are related to any type of heating of gas
and that are related to any type of constant volume."

This yields no concepts. He then removes the sub-expression containing heating of gas, and then still gets no concepts. Subsequently, he removes the sub-expression containing constant volume and re-adds heating of gas. This yields the specification as before, and thus also the same four concepts. He then inspects the explanation for each concept. Next, he opens the Quick Browser, and finds the relationship between constant volume and isochoric process. After closing the Quick Browser he creates a new specification, in which constant volume is substituted by isochoric process. The specification is:

"I want the concepts that are any type of regenerative thermal cycle
and that have relation of type comprise to any type of isothermal expansion of gas
and that are related to any type of heating
and that are related to any type of isochoric process."

He does not wait for the solution (due to the large number of sub-expressions the solving process lasts long), and erases after approximately half a minute the subexpression containing heating. Then, he again does not wait for the answer, but tries after less than a minute the Quick Browser and looks for isochoric process. He then closes the Quick Browser, and specifies:

"I want the concepts that are any type of regenerative thermal cycle
and that have a relation of type comprise to any type of isothermal expansion of gas."

This yields the same four concepts as obtained before. He then checks these concepts in the explanation facility and the Quick Browser, and selects the three concepts of the solution proper.

D.3 Solutions provided by the test persons

Below, the concepts identified by the test persons are shown.

Test person 1

Task 1	**Task 2**	**Task 3**
open Brayton cycle	Philips cycle	heat pipe
closed Brayton cycle	Stirling cycle	bellows type heat pipe
Brayton cycle	Vuilleumier cycle	flexible heat pipe
Ackerett-Keller cycle		thermal switch heat pipe
Ericsson cycle		

Test person 2

Task 1	**Task 2**	**Task 3**
open Brayton cycle	Philips cycle	flexible heat pipe
closed Brayton cycle	Stirling cycle	thermal switch heat pipe
Ackerett-Keller cycle	Vuilleumier cycle	
Ericsson cycle		

Test person 3

Task 1	**Task 2**	**Task 3**
Ericsson cycle	Philips cycle	bellows type heat pipe
	Stirling cycle	
	Vuilleumier cycle	

Test person 4

Task 1	**Task 2**	**Task 3**
Brayton cycle	Philips cycle	bellows type heat pipe
Ackerett-Keller cycle	Stirling cycle	
	Vuilleumier cycle	

Test person 5

Task 1	**Task 2**	**Task 3**
Brayton cycle	Philips cycle	bellows type heat pipe
Ackerett-Keller cycle	Stirling cycle	
Ericsson cycle	Vuilleumier cycle	

Test person 6

Task 1	**Task 2**	**Task 3**
Brayton cycle	Philips cycle	bellows type heat pipe
Ackerett-Keller cycle	Stirling cycle	
Ericsson cycle	Vuilleumier cycle	

Appendix E
The transformation experiment

E.1 Tasks and relevance criteria

Below, we provide (1) the three tasks for the experiment of the transformation step of Dipe-D's query-formulation process, and (2) the relevance criteria for the documents identified in this experiment, for each of the three tasks.

The tasks were completed by the test persons in the experiment, as described in Chapter 8. The relevance criteria were used for judging the relevance of the documents identified.

Task 1

The task is: "We distinguish three types of regenerative thermal cycles: the Stirling cycle, the Vuilleumier cycle, and the Ericsson cycle. A Philips cycle is a subtype of the Stirling cycle. The sun yields solar energy, and is a celestial body. Make a query for finding documents about regenerative thermal cycles that are powered by sunlight. Try to find relevant documents only".

The judges consider an abstract relevant only if the cycle and details of the cycle are mentioned in the abstract. Moreover, it must be mentioned that it is possible that the cycle is powered by solar energy. Thus, for instance, an abstract is considered irrelevant if it mentions that the subject of the abstract (e.g., a heat-exchanger) is possibly used in combination with a Stirling cycle, and does not (explicitly) mention solar energy.

Task 2

The task is: "There exist many types of heat pipes. The common type is the standard heat pipe, often referred to simply as heat pipe. Some heat pipes contain a bellows, some heat pipes with bellows are named bellows type heat pipes. Make a query for finding documents about heat pipes containing a bellows. Try to find relevant documents only".

An abstract is regarded relevant only if it has as main subject a heat pipe, that somehow contains a bellows.

Task 3

The task is: "Make a query for finding documents about thermosyphons (also called thermal syphons) meant/used for heating something. Try to find relevant documents only".

An abstract is regarded relevant only if it learns something about possible constructions of a thermosyphon.

The thermosyphon may be used for heating, or cooling, or for both. Although the task mentions cooling only, due to the physical operating principles of a heat pipe, cooling by a thermosyphon always co-occurs with heating by the thermosyphon and with mass-transport. It therefore suffices if either heating or cooling is mentioned.

E.2 Search results

The results of the matching of the queries formulated are shown in Table E.1. They include, per pair of person and task (i.e., per query): the number of matched documents, the number of relevant documents, the recall, the precision, and the effectiveness. The queries formulated are reported on the WWW[89].

Person (#)	Task (#)	Matched (#)	Relevant (#)	Precision (%)	Recall (%)	Effectiveness (%)
1	1	0	0	0	0	0
	2	12	10	83.33	100	90.9
	3	65	34	52.12	72.34	60.7
2	1	0	0	0	0	0
	2	28	10	35.71	100	52.6
	3	18	10	55.56	21.28	30.8
3	1	18	3	16.67	60	26.1
	2	12	10	83.33	100	90.9
	3	23	12	52.17	25.53	34.3
4	1	5	2	40	40	40
	2	2	2	100	20	33.3
	3	29	15	51.72	31.91	39.5
5	1	0	0	0	0	0
	2	2	2	100	20	33.3
	3	50	32	64	68.09	33.0
6	1	0	0	0	0	0
	2	12	10	83.33	100	90.9
	3	68	33	48.53	70.21	57.4
7	1	20	2	10	40	16
	2	2078	–	~0	100	0
	3	70	47	67.14	100	80.3
8	1	2	0	0	0	0
	2	1	1	100	10	18.2
	3	1	1	100	2.13	4.2
Dipe-D	1	0	0	0	0	0
	2	12	10	83.33	100	90.9
	3	23	12	52.17	25.23	34.4
Dipe-D	1	3	0	0	0	0
wre	2	18	10	55.56	100	71.4
	3	24	12	50	25.53	33.8
Dipe-D	1	8	3	37.5	60	46.2
sre	2	18	10	55.56	100	71.4
	3	24	12	50	25.53	33.8

TABLE E.1: *Search results per person and task.*

89 http://www.cs.unimaas.nl/~pol/transformation_tasks.htm

References

Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications.* Vol. 7, No. 1, Pp. 33-59.

Aboud, M., Chrisment, C., Razouk, R., Sedes, F., Soule-Dupuy, C. (1993). Querying a hypertext information retrieval system by the use of classification. *Information Processing & Management.* Vol. 29, No. 3, Pp. 387-396.

Aboud, M., Razouk, R., Sedes, F, Soule-Dupuy, C. (1991). Classification and Information Retrieval in Hypertext Systems. *Procs. of a conference on intelligent text and image handling "RIAO'91"* (Ed. A. Lichnerowicz), Pp. 103-118. Elsevier, Amsterdam, The Netherlands.

Akmajian, A., Demers, R.A., and Harnish, R.M. (1979*). Linguistics: an introduction to language and communication.* MIT Press, Cambridge, MA.

Arents, H.C. and Bogaerts, W.F.L. (1993). Concept-Based Retrieval of Hypermedia Information: from term indexing to semantic hyperindexing. *Information Processing & Management.* Vol. 29, No. 3, Pp. 373-386.

Audi, R. (1998). *Epistemology; a contemporary introduction to the theory of knowledge.* Routledge, London, UK.

Ballesteros L. and Croft, W.B. (1997). Phrasal Translation and Query Expansion Techniques for Cross-Language Information Retrieval. *Procs. of the 20th Annual International* ACM-SIGIR *Conference on Research and Development in Information Retrieval.* (Eds. N. Belkin, A.D. Narasimhalu, and P. Wilett) Pp. 84-91. ACM, New York, NY.

Belkin, N.J. and Croft, W. (1992). Information Filtering and Information Retrieval: two sides of the same coin? *Communications of the* ACM. Vol. 35, No. 12, Pp. 29-38.

Belkin, N.J., Marchetti, P.G., and Cool, C. (1993). BRAQUE: Design of an Interface to Support User Interaction in Information Retrieval. *Information Processing & Management.* Vol. 29, No. 3, Pp. 325-344.

Berger, C.R. and Chaffee, S.H. (Eds.) (1987). *Handbook of communication science.* Sage Publications, Newbury Park, UK.

Berlo, D.K. (1960). *The process of communication : an introduction to theory and practice.* Holt, Rinehart and Winston, New York, NY.

Berners-Lee, T., Cailliau, R., Groff, J.F., and Pollermann, B. (1992). World-Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy.* Vol. 1, No. 2, Pp. 52-58. Meckler, Westport CT.

Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H.F., Secret, A. (1994). The World-Wide Web. *Communications of the ACM*. Vol. 37, No. 8, Pp. 76-82.

Blair, D. C. (1996). STAIRS redux: Thoughts on the Stairs Evaluation, ten years after. *Journal of the American Society for Information Science*. Vol. 47, No. 1, Pp. 4-22.

Blair, D. C. and Maron, M.E. (1985). An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Communications of the ACM*. Vol. 28, No. 3, Pp. 289-299.

Bookstein, A. (1980). Fuzzy requests: An approach to weighted Boolean searches. *Journal of the American Society of Information Science*. Vol. 31, No. 4, Pp. 240-247.

Bookstein, A. (1981). A comparison of two systems of weighted Boolean retrieval. *Journal of the American Society for Information Science*. Vol. 32, No. 4, Pp. 275-279.

Borgida, A., Brachman, R. J., McGuinness, D. L., and Resnick, L. A. (1989). CLASSIC*: A Structural Data Model for Objects.* Proceedings of ACM SIGMOD International Conference on Management of Data. Pp. 59-67.

Borgida, A. and Patel-Schneider, P. F. (1994). A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *Journal of Artificial Intelligence Research*. Vol. 1, Pp. 277-308.

Borst, W.N., Top, J.L., and Akkermans, J.M. (1994). Physical Systems Ontology. In (editor N.J.I. Mars): *Working Papers European Conference on Artificial Intelligence ECAI'94 Workshop on Implemented Ontologies.* Pp. 47-80. Amsterdam, The Netherlands.

Boughanem, M. (1992). *Les systèmes de recherche et d'informations. D'un modèle classique à un modèle connexioniste.* Ph.D thesis. Univ. Paul Sabatier de Toulouse, Toulouse, France.

Brachman, R.J. (1979). On the Epistemological Status of Semantic Networks. *Associative Networks: Representation and Use of Knowledge by Computers* (Ed. N.V. Findler), Pp. 3-50. Academic Press, New York, NY.

Brachman, R.J. (1983). What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer*. Vol. 16, No. 10.

Brachman, R.J. (1985). I lied about the trees. *AI Magazine*. Vol. 6, No. 3, pp. 80-93.

Brachman, R., Fikes, R., and Levesque, H. (1983). KRYPTON: A Functional Approach to Knowledge Representation. *IEEE Computer*. Vol. 16, No. 10, Pp. 67-73.

Brachman, R.J. and Schmolze, J.G. (1985). An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*. Vol. 9, No. 2, Pp. 171-216.

Braspenning, P.J. (1989). A Framework for Modelling Complex Objects. *Proceedings of Hierarchical Object-Oriented Design.* Brunel Science Park, Uxbridge, UK.

Bratko, I. (1990). *Prolog Programming for Artificial Intelligence.* (2nd edition) Addison-Wesley, Reading, MA.

Bryan, J. (1995). Compression Scorecard. *Byte*. Vol. 20, No. 5, Pp. 107-110.

Buchanan, B.G. Shortliffe, E.H. (1984). *Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, MA.

Buckley, C., Salton G., and Yu, G.T. (1982). An evaluation of term dependence models in information retrieval. *Research and Development in Information Retrieval.* (Lecture notes in Computer Science. Vol. 146.) Springer, Berlin, Germany.

Buckley, C., Salton, G., Allan, J., and Singhal, A. (1995). Automatic Query Expansion Using SMART: TREC 3. *Overview of the Third Text Retrieval Conference* (*TREC-3*). Pp. 69-80.

Buell, D.A. and Kraft, D.H. (1981). Threshold values and Boolean retrieval systems. *Information Processing and Management*. Vol. 17, No. 3, Pp. 127-136.

Bunge, M. (1973). *Treatise on Basic Philosophy; volume 1; Semantics I: Sense and Reference.* Reidel Publishing Company, Dordrecht, the Netherlands.

Bunge, M. (1974). *Treatise on Basic Philosophy; volume 4; Systems.* Reidel Publishing Company,

Dordrecht, the Netherlands.

Callan, J.P., Croft, W.B., and Harding, S.M. (1992). The INQUERY retrieval system. *Procs. of the Third International Conference on Database and Expert Systems Applications* (Eds. A.M. Tjoa and I. Ramos). Pp. 78-83.

Carbonell, J.G., Yang, Y., Frederking, R.E., Brown, R.D., Geng, Y., and Lee, D. (1997). Translingual Information Retrieval: A Comparative Evaluation. *Procs. of the Fifteenth International Joint Conference on Artificial Intelligence.* (*Vol. 1*) Pp. 708- 714. Nagoya, Japan.

Chakrabarti, S., Dom, B.E., Gibson, D., Kumar, R., Raghavan, R., Rajagopalan, S., and Tomkins, A. (1998). *Experiments in Topic Distillation.* IBM Almaden Research Center, San Jose, CA.

Chen, H., Lynch, K.J., Basu, K., and Ng, T. (1993). Generating, integrating, and activating thesauri for concept-based document retrieval. *IEEE Expert.* Vol. 8, No. 2, Pp. 25-34.

Chen, P-W. and Wang, S-K. (1997). Knowledge-based Multimedia Information Retrieval in Hyperspace. *Telematics and Informatics.* Vol. 14, No. 1, Pp. 27-50.

Cherry, C. (1957). *On human communication; a review, a survey, and a criticism.* John Wiley & Sons. New York, NY.

Chong, A. (1989). Topic: a concept-base document retrieval system. *Library software review.* Vol. 8, Pp. 281-284.

Chudáçek, J. (1984). Niet-grammaticale verwerking van natuurlijke talen in computers. *Informatie.* Vol. 28, No. 8, Pp. 594-599. (In Dutch) {Non-grammatical processing of natural languages by computers.}

Cohen, P.R. and Kjeldsen, R. (1987). Information Retrieval by Constrained Spreading Activation in Semantic Networks. *Information Processing & Management.* Vol. 23, No. 4, Pp. 255-268.

Coll, R. A., Coll, J.H., Thakur, G. (1994). Graphs and tables, a four-factor experiment. *Communications of the ACM.* Vol. 37, No. 4, Pp. 78-87.

Collins, A.M. and Loftus, E.F. (1975). A spreading activation theory of semantic processing. *Psychological Review.* Vol. 82, No. 6, Pp. 407-428.

Concise (1995). *The Concise Oxford Dictionary of Current English.* 9th ed. (Ed. Allen, R.E.) Oxford University Press Inc., New York, NY.

Copi, I.M. (1982). *Introduction to logic.* (Sixth edition) MacMillan Publishing Co, New York, NY.

Creput, J-C. and Caron, A. (1997). An Information System Using a New Neural Network Model. *Cybernetica.* Vol. 40, No. 2, Pp. 127-139.

Crestani, F. (1994). Domain Knowledge Acquisition for Information Retrieval Using Neural Networks. *Int. Journal of Applied Expert Systems.* Vol. 2, No. 2, Pp. 101-115.

Crestani, F. (1997). Application of Spreading Activation Techniques in Information Retrieval. *Artificial Intelligence Review.* Vol. 11, Pp. 453-482.

Crestani, F. and Lalmas, M. (Eds.) (1996). *Second Workshop on Information Retrieval, Logic and Uncertainty.* Techn. Report, TR-1996-29. Dept. of Computer Science, Univ. of Glasgow, Glasgow, Scotland.

Crestani, F. and Van Rijsbergen, C.J. (1995). Probability kinematics in information retrieval. *Procs. of the 18th ACM-SIGIR Conference on Research and Development in Information Retrieval.* Pp. 291-299. (Eds. E.A. Fox, P.Ingwersen, and R. Fidel). ACM, New York, NY.

Croft, W.B. (1993). Knowledge-Based and Statistical Approaches to Text Retrieval. *IEEE Expert.* Vol. 8, No.2, Pp. 8-12.

Croft, W.B. and Thompson, R.H. (1987). I^{3}R: A New Approach to the Design of Document Retrieval Systems. *Journal of the American Society for Information Science,* Vol. 38, No. 6, Pp. 389-404.

Crouch, D., Crouch, C.J., and Andreas, G. (1989). The use of cluster hierarchies in hypertext information retrieval. *HYPER89 Proceedings of ACM Hypertext '89 Conference* (eds. F. Halasz and N. Meyrowitz), Pp. 225-237. ACM Press, Pittsburgh, PA.

Date, C.J. (1990). *An introduction to Database Systems; volume 1.* (5th edition) Addison-Wesley

Publishing Company, Readings, MA.

De Bra, P.M.E. and Post, R.D.J. (1995). Information retrieval in the World Wide Web: Making client-based searching feasible. *Computer networks and ISDN systems: the international journal of computer and telecommunications networking.* Vol. 27, No. 2, Pp. 183-192.

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., and Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science.* Vol. 1, No. 6, Pp. 391-407.

De Jong, D.J. (1990). *Method of displaying navigation data for a vehicle in an image of the vehicle environment, a navigation system for performing the method, and a vehicle comprising a navigation system.* European Patent Application No. 0 406 946 A1.

Dietz, J.L.G. (1993). *Theoretical Foundations for Modelling Facilities.* Research Memorandum 93-037. University of Limburg, Faculty of Economic Sciences, Maastricht, The Netherlands.

Dietz, J.L.G. (1994). *A universal, ontology-free conceptual modelling technique.* Research Memorandum. Delft University of Technology, Delft, The Netherlands.

Dietz, J.L.G. (1996). *Wat doen computers als ze iets zeggen?* Inaugural Address. Delft University of Technology, Delft, The Netherlands. (In Dutch.) {What are computers doing when they are saying something?}

Dietz, J.L.G. (1999). Personal communication.

Dietz, J.L.G., Van der Pol, R.W., and Wiesman, F. (1997). The ARCHIMEDES Network System. *Journal of Intelligent Information Systems.* Vol. 8, No. 1, Pp. 77-101. Also: Internal report SKBS/B3.A/94-08.

Doets, H.C. and Troelstra, A.S. (1983). *Inleiding in de Wiskunde en Verzamelingenleer.* Reader. Universiteit van Amsterdam, Amsterdam, The Netherlands. (In Dutch) {Introduction to Mathematics and Set Theory.}

Dong, X. and Su, L.T. (1997). Search Engines on the World Wide Web and Information Retrieval from the Internet: a Review and Evaluation. *Online & CD-ROM Review.* Vol. 21, No. 2, Pp. 67-81.

Doszkocs, T.E., Reggia, J., and Lin, X. (1990). Connectionist Models and Information Retrieval. *Annual Overview of Information Science and Technology.* Vol. 25, Pp. 209-260.

Dunn, P.D. and Reay, D.A. (1994). *Heat pipes.* (Fourth Edition) Pergamon, Oxford, UK.

Eades, P. (1984). *A heuristic for graph drawing.* Congressus Numerantium. Vol. 42, Pp. 149-160.

Eekels, J. and Roozenburg, N.F.M. (1979). *Beknopte Methodologie voor Ingenieurs.* Reader. Delft University of Technology, Delft, The Netherlands. (In Dutch) {Concise Methodology for Engineers.}

Efthimiadis, E.N. (1996). Query Expansion. *Annual Review of Information Science and Technology.* Vol. 31, Pp. 121-187. (Ed. M.E. Williams)

Evans, D.A., Ginther-Webster K., Hart M., Lefferts R.G., and Monarch I.A. (1991). Automatic indexing using selective NLP and first-order thesauri. *Procs of a conference on intelligent text and image handling "RIAO'91"* (Ed. A. Lichnerowicz). Pp. 624-643. Elsevier. Amsterdam, The Netherlands.

Feiten, B. and Günzel, S. (1994). Automatic Indexing of a Sound Database Using Self-organizing Neural Nets. *Computer Music Journal.* Vol. 18, No. 3, Pp. 53-65.

Fiske, J. (1982). *Introduction to communication studies.* Methuen, London, UK.

Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., and Yanker, P. (1995). Query by Image and Video Content: The QBIC System. *Computer.* Vol. 28, No. 9, Pp. 23-32.

Fodor, J.A. (1998). *Concepts: where cognitive science went wrong.* Clarendon Press, Oxford, UK.

Fowler, R.H., Fowler, W.A., and Williams, J.L. (1998*). Document Explorer Visualizations of WWW Document and Term Spaces.* Report Dept. of Computer Science, Univ. of Texas. Pan American Edinburg, TX.

Frakes, W.B. and Beaza-Yates, R. (1992). *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall, Englewood Cliffs, NJ.

Frei, H.P. and Stieger, D. (1995). The use of semantic links in hypertext information retrieval. *Information*

processing & Management. Vol. 31, No. 1, Pp. 1-13.

Gaizauskas, R. and Wilks, Y. (1998). Information Extraction: Beyond Document Retrieval. *Journal of Documentation*. Vol. 54, No. 1, Pp. 70-105.

Garner, R. (1999). Features - TECHNOLOGY - The E-commerce Connection - The Internet gives companies new ways to solidify customer relationships, reduce costs, and build profits. *Sales & marketing management*. Vol. 151, No. 1, Pp. 40-47.

Gazzaniga, M.S. (Ed.) (1995). *The Cognitive Neurosciences*. MIT Press, Cambridge, MA.

Gruber, T.R. (1991). The role of common ontology in achieving sharable, reusable knowledge bases. *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*. (Eds. J.A. Allen, R. Fikes, and E. Sandewall) Pp. 601-602. Morgan Kaufmann. Cambridge, MA.

Gruber, T. R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition*, Vol. 5, No. 2, Pp. 199-220.

Guarino, N. (1992). Concepts, attributes and arbitrary relations; Some linguistic and ontological criteria for structuring knowledge bases. *Data & Knowledge Engineering*. Vol. 8, Pp. 249-261.

Guarino, N., Masolo, C., and Vetere, G. (1999). OntoSeek: Using Large Linguistic Ontologies for Accessing On-Line Yellow Pages and Product Catalogs. *IEEE Intelligent Systems*. Vol. 14, No. 2, Pp. 70-80.

Guha, R.V. and Lenat, D.B. (1991). CYC: A Midterm Report. *AI Magazine*. Vol. 11, No. 3, Pp. 32-59.

Guha, R.V., and Lenat, D.B. (1994). Enabling agents to work together. *Communications of the ACM*. Vol. 37, No. 7, Pp. 127-142.

Gupta, A, and Jain, R. (1997). Visual information retrieval. *Communications of the ACM*. Vol. 40, No. 5, Pp.71-79.

Hahn, U. and Reimer, U. (1988). Automatic Generation of Hypertext Knowledge Bases. *SIGOIS Bulletin (ACM)*. Vol. 9.

Hamel, R. (1990). *Over het denken van de architect*. Krips Repro, Meppel, The Netherlands. (In Dutch) {The thinking of architects[90].}

Hayes, P. (1977). *Some association-based techniques for lexical disambiguation by machine*. Ph.D. Dissertation, published as Tech. Report. 25. Dept. of Computer Science, Univ. of Rochester, NY.

Hemmje, M. (1994). A 3D Based User Interface for Information Retrieval Systems. *Lecture notes in computer science*. Vol. 871, Pp. 194-209.

Herbst, K. (1996). Webfest IV - The fourth World-Wide Web Conference in Boston was abuzz with Web talk. Hot toppics included Web commerce, Java, VRML, and HTML extensions. *Internet world*. Vol. 7, No. 3, Pp. 22-27.

Hertz, J., Krogh, A., and Palmer, R.G. (1991). *Introduction to the theory of neural computation*. Addison-Wesley, Reading, MA.

Hirst, G. (1988). Resolving lexical ambiguity computationally with spreading activation and polaroid words. *Lexical Ambiguity Resolution*. (Eds. S. Small, G. Cottrell, and M. Tannenhaus). Morgan Kaufmann, Palo Alto, CA.

Honderich, T. (Ed.) (1995). *The Oxford Companion to Philosophy*. Oxford University Press, Oxford, UK.

Hoogeveen, M.J. (1995). *The Viability of Multimedia Retrieval Systems for Marketing and Sales*. Ph.D. Thesis. Delft University of Technology, Delft, The Netherlands.

Hoogeveen, M.J. (1999). Altavista op 1 met stip. *Computable*. Vol. 32, No. 20, Pp. 33-35. (In Dutch.)

---

90 The Dutch title is a variation on *Over het denken van den schaker* (1946), by A.D. de Groot. The latter was published in English as *Thought and Choice in Chess* (1965). In analogy, the English title could become Thought and Choice in Architecture.

{Altavista a climber on 1.}

Horn, P. M. (1999). Information Technology Will Change Everything. *Research Technology Management.* Vol 42, No. 1, Pp. 42 – 48.

Howe, A.E. and Dreilinger, D. (1997). SavvySearch; a Metasearch Engine That Learns Which Search Engines to Query. *AI Magazine.* Vol. 18, No. 2, Pp. 19-25.

Hull, D. (1993). Using statistical testing in the evaluation of retrieval experiments. *Procs. of the 16th International Conference on Research and Development in Information Retrieval.* (Eds. R. Korfhage, E.M. Rasmussen, and P. Willett) Pp. 338-349. ACM, New York, NY.

Hull, D.A. (1997). Using Structured Queries for Disambiguation in Cross-Language Information Retrieval. *AAAI Spring Symposium on Cross-Language Text and Speech Retrieval Electronic Working Notes.* URL: http://www/clis.umd.edu/dlrg/filter/sss/papers/ hull3.ps.

Hurt, C.D. (1998). Nonmonotonic logic for use in information retrieval: an exploratory paper. *Information Processing & Management.* Vol. 34, No. 1, Pp. 35-41.

Ingwersen, P. (1996). Cognitive perspectives of information retrieval interaction: elements of a cognitive IR theory. *Journal of Documentation.* Vol. 52, No. 1, Pp. 3-50.

ISO 2382-1 (1993). *ISO 2382-1 Dataprocessing vocabulary*. International Organization for Standardization, Geneva, Switserland.

ISO 2788 (1986). *ISO 2788, Documentation - Guidelines for the establishment and development of monolingual thesauri.* Geneva: International Organization for Standardization.

Jansen, B.J., Spink, A., Bateman, J., and Saracevic, T. (1998). Real life Information Retrieval: A Study Of User Queries On The Web. *SIGIR forum: a publication of the ACM Special Interest Group on Information Retrieval.* Vol. 32, No. 1, Pp. 5-17.

Järvelin, K., Kristensen, J., Niemi, T., Sormunen, E., and Keskustalo, H. (1996). A Deductive Data Model for Query Expansion. *Proceedings of the 19th ACM Annual International ACM-SIGIR conference.* Pp. 235–249. (Eds. H.–P. Frei, D. Harman, P. Schäuble, and R. Wilkinson) ACM, New York, NY.

Jing, Y. and Croft, W.B. (1994). An Association Thesaurus for Information Retrieval. Dept. of Computer Science, Univ. of Massachusetts at Amherst, Amherst, MA.

Jones, S. (1993). A thesaurus data model for an intelligent retrieval system. *Journal of Information Science.* Vol. 19, Pp. 167-178.

Jones, S., Gatford, M., Robertson, S., Hancock-Beaulieu, M., Secker, J., Walker, S. (1995). Interactive Thesaurus Navigation: Intelligence Rules OK? *Journal of the American Society for Information Science.* Vol. 46, No. 1, Pp. 52-59.

Kekäläinen, J. and Järvelin, K. (1998). The impact of query structure and query expansion on retrieval performance. *Procs. of the 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* (Eds. W.B. Croft, A. Moffat, C.J. van Rijsbergen, R. Wilkinson, and J. Zobel) Pp. 130-137. ACM, New York, NY.

Kekäläinen, J. (1999). *The Effects of Query Complexity, Expansion and Structure on Retrieval Performance in Probabilistic Text Retrieval.* Ph.D. thesis. Univ. of Tampere, Tampere, Finland

Kent, A., Berry M., Luehrs, F.U., and Perry, J.W. (1955). Machine Literature Searching VIII. Operational Criteria for Designing Information Retrieval Systems. *American Documentation.* Vol. 6, No. 2, Pp. 93-101.

Kim, Y.W. and Kim, J.H. (1990). A Model of Knowledge-Based Information Retrieval with Hierarchical Concept Graph. *Journal of Documentation.* Vol. 46, No. 2, Pp. 113-136.

Kochen, M. (1974). *Principles of Information Retrieval.* Melville Publishing Company, Los Angeles, CA.

Koenen, R. (1999). *MPEG-4*; Multimedia for our time. *IEEE Spectrum.* Vol. 36, No. 2, Pp. 26-33.

Koster, M. (1994). *ALIWEB* - Archie-Like Indexing in the *WEB. Computer networks and ISDN systems: the international journal of computer and telecommunications networking.* Vol. 27, No. 2, Pp. 175-182.

Kristensen[91], J. (1993). Expanding end-user's query statements for free text searching with a search-aid thesaurus. *Information Processing & Management.* Vol. 29, No. 6, Pp. 733-744.

Krovetz, R., and Croft, W.B. (1992). Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems.* Vol. 10, No. 2, Pp. 115-141.

Kwok, K.L. (1990). Experiments with a component theory of probabilistic information retrieval based on single terms as document components. *ACM Transactions on Information Systems.* Vol. 8, No. 4, Pp. 363-386.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things; What Categories Reveal about the Mind.* The University of Chicago Press, Chicago, IL.

Lalmas, M. (1998). Logical Models in Information Retrieval: Introduction and Overview. *Information Processing & Management.* Vol. 34, No. 1, Pp. 19-33.

Langefors, B. and Samuelson, K. (1976). *Information and Data in Systems.* Petrocelli, New York, NY.

Lee, J.H. (1994). Properties of Extended Boolean Models in Information Retrieval. *Procs. of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* (Eds. W. B. Croft and C.J. van Rijsbergen) Pp. 182-190. ACM, New York, NY.

Lee, J.H., Kim, M.H., and Lee, Y.J. (1993). Information Retrieval Based on Conceptual Distance in IS-A Hierarchies. *Journal of Documentation.* Vol. 49, No. 2, Pp. 188-207.

Lee, J.H., Kim, M.H., and Lee, Y.J. (1994). Ranking Documents in Thesaurus-Based Boolean Retrieval Systems. *Information Processing and Management.* Vol. 30, No. 1, Pp. 79-91.

Lenat, D.B. (1995). CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM.* Vol. 38, No. 11, Pp. 33-38.

Lenat, D.B. (1998). *The Dimensions of Context-Space.* Report Cycorp, Austin, TX.

Lenat, D.B. and Guha, R.V. (1990). *Building large knowledge-based systems: representation and inference in the Cyc Project.* Addison-Wesley Publishing Company, Inc., Reading, MA.

Lenat, D.B. and Guha, R.V. (1991). The evolution of CycL: The CYC Representation Language. *Sigart Bulletin: Special Issue on Implemented Knowledge Representation and Reasoning Systems.* Vol. 2, No. 3, Pp. 84-87.

Lenat, D., Prakash, M., and Sheperd, M. (1986). CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks. *AI Magazine.* Vol. 6, No. 4, Pp. 65-85.

Lin, X., Soergel, D., and Madchioni, G. (1991). A Self-Organizing Semantic Map for Information Retrieval. *Proc. Of the 14th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* (Eds. A. Bookstein, Y. Chiaramella, G. Salton, and V.V. Raghavan) Pp. 262-269. ACM, New York, NY.

Lindberg, D.A.B., Humphreys, B.L., and McCray, A.T. (1993). The unified medical language system. *Yearbook of Medical Informatics 1993.* Pp. 41-51. Schattauer, Stuttgart, Germany.

Lindsay, P. H. and Norman, D. A. (1972). *Human information processing.* Academic Press, New York, NY.

Lundeberg, M., Goldkuhl, G., Nilsson, A. (1981). *Information Systems development: a systematic approach.* Prentice Hall, Englewood Cliffs, NJ.

MacGregor, R.M. and Bates, R. (1987). *The Loom knowledge representation language.* ISI RS-87-188. Information Sciences Institute, University of Southern California, CA.

Maida, A.S. and Shapiro, S.C. (1982). Intensional Concepts in Propositional Semantic Networks. *Readings in knowledge representation.* (Eds. R. J. Brachman and H. J. Levesque) Pp. 169-189. Kaufmann, Los Altos, CA.

Matthijssen, L. (1999). *Interfacing between Lawyers and Computers; an Architecture for Knowledge-based*

91 Kekäläinen and Kristensen name the same person.

Interfaces to Legal Databases. Ph.D. thesis. Katholieke Universiteit Brabant, Tilburg, The Netherlands.

McCune, B.P., Tong, R.M, Dean, J.S., Shapiro, D.G. (1985). RUBRIC: A System for Rule-Based Information Retrieval. *IEEE Transactions on Software Engineering.* Vol. 11, No. 9, Pp. 939-945.

McGraw-Hill (1997). *Dictionary of Engineering.* Mc-Graw-Hill, New York, NY.

McMath, C.F., Tamaru, R. S., and Rada, R. (1989). A graphical thesaurus-based information retrieval system. *International Journal of Man-Machine Studies.* No. 31, Pp.121-147.

Miller, G.A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM.* Vol. 38, No. 11, Pp. 39- 41.

Mitra, M., Singhal, A., and Buckley, C. (1998). Improving Automatic Query Expansion. *Procs. Of the 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* (Eds. W.B. Croft, A. Moffat, C.J. van Rijsbergen, and J. Zobel) Pp. 206-214. ACM, New York, NY.

Mizzaro, S. (1998). How many relevances in information retrieval? *Interacting with Computers.* Vol. 10, Pp. 303-320.

Mothe, J. (1994). *Modèle connexionniste pour la recherche d'information.* Thèse de Doctorat de l'université Paul Sabatier de Toulouse (sciences). Toulouse, France.

Nachtmann, L. (1999). *DVD* – Laufwerke im Test: Besser als *CD-ROM*-Drives. *Chip.* No. 5, Pp. 110-131.

Naur, P., Backus, J.W., Bauer, F.L., Green, J., Katz, C., McCarthy, J., Perlis, A.J., Rutishauer, Samelson, K., Vauquois, B., Wegstein, J.H., Van Wijngaarden, A., and Woodger, M. (1963). Revised report on the algorithmic language Algol 60. *Communications of the ACM.* Vol. 6, No. 1, Pp. 1-17.

Navon, D. (1984). Resources: a theoretical soupstone? *Psychological Review.* Vol. 91, No. 2, Pp. 216-234.

Neisser, U. (Ed.) (1987). *Concepts and Conceptual Development: Ecological and intellectual factors in categorization.* Cambridge University Press, Cambridge, NY.

Newell, A. (1975). *Unified Theories of Cognition.* W.W. Norton & Company, New York, NY.

Newell, A. and Simon, H.A. (1972). *Human Problem Solving.* Prentice-Hall, Inc., Englewood Cliffs, NJ.

Notes, G.R. (1995). On the nets - Searching the world-wide web: Lycos, WebCrawler and more. *Online: the magazine of online information systems.* Vol. 19, No. 4, Pp. 48-53.

Nijssen, G.M. (1993). *Universele Informatiekunde.* PNA Publishing BV. Beutenaken, Belgium. (In Dutch) {Universal Information Science.}

Nijssen, G.M. and Halpin, T.A. (1989). *Conceptual Schema and Relational Database Design: A Fact Oriented Approach.* Prentice Hall, Sydney.

Ogden, C.K. and Richards, I.A. (1923). *The meaning of meaning.* Routledge and Kegan Paul Ltd, London, UK.

Ollongren, A. and Herik, H.J. van den (1989). *Filosofie van de Informatica.* Nijhoff, Leiden, The Netherlands. (In Dutch) {Philosophy of Computer Science.}

Parsaye, K., Chignell, M., Khoshafian, S. and Wong, H. (1989). *Intelligent Databases: Object-Oriented, Deductive Hypermedia Technologies.* John Wiley & Sons, New York, NY.

Paijmans, H. (1999). *Explorations in the Document Vector Model of Information Retrieval.* Ph.D. Thesis. Catholic University of Brabant, Tilburg, The Netherlands.

Pfeiffer, U. (1995). HTTPS älterer Bruder. WAIS: Inhaltsorientierte Suche im Internet. *Wissen.* Vol. 9, No. 1, Pp. 120-127.

Pietrosanti, E. and Graziadio, B. (1997). Artificial intelligence and legal text management: tools and techniques for intelligent document processing and retrieval. In: *Natural Language Processing: Extracting Information for Business Needs.* Pp. 277-291. Unicom Seminars Ltd, London, UK.

Pirkola, A. (1998). The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. *Proceedings of the 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. (Eds. W.B. Croft, A. Moffat, C.J. van Rijsbergen, R. Wilkinson, and J. Zobel) Pp. 55-63.* ACM, New York, NY.

Plugge, L.A. (1997). *Personal communication.* See also: *Evince; A neuropsychiatric Desktop Expert System for the Diagnosis of Dementia.* Ph.D. Thesis, 1992. Rijkskuniversiteit Limburg, Maastricht, The Netherlands.

Prak, N.L. (1979). *Vorm en betekenis.* Delftse Universitaire Pers. Delft, The Netherlands. (In Dutch) {Form and meaning.}

PTO (1996). http://patents.cnidr.org/pto/classes/us/62/62.html

Putnam, H. (1973). Meaning and reference. *The Journal of Philosophy.* Vol. 70, No. 19, Pp. 699-711.

Qiu, Y. and Frei, H.P. (1993). Concept-based Query Expansion. *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* (Eds. R. Korfhage, E.M. Rasmussen, and P. Willett) Pp. 160-169. ACM, New York, NY.

Quillian, M. R. (1966). *Semantic Memory.* Unpublished doctoral dissertation. Carnegie Institute of Technology. Also report AFCRL-66-189, Bolt, Beranek, and Newman, Cambridge, MA.

Quillian, M.R. (1968). Semantic memory. *Semantic information processing.* (Ed. M. Minsky), Pp. 216-270. MIT Press, Cambridge, MA.

Quine, W. v. O. (1960). *Word and object.* MIT Press. Cambridge, MA.

Rada, R. and Bicknell, E. (1989). Ranking Documents with a Thesaurus. *Journal of The American Society for Information Science.* Vol. 40, No. 5, Pp. 304-310.

Rada, R., Bicknell, E., Humphrey, S., Suh, A., and Coccia, C. (1985). Relevance on a biomedical classification structure. *Proceedings Expert Systems in Government Conference*, Pp. 532-537. IEEE Computer Society Press.

Rada, R., Mili, H., Bicknell, E., and Blettner, M. (1989). Development and Application of a Metric on Semantic Nets. *IEEE Transactions on Systems, Man, and Cybernetics.* Vol. 19, No. 1, Pp. 17-30.

Ram, A. (1999). Pentium III. *Personal Computer World.* No. 4, Pp. 122- 128.

Rama, D.V. and Srinivasan, P. (1993). An investigation of content representation using text grammars. *ACM Transactions on Information Systems.* Vol. 11, No.1, Pp. 51-75.

Reid, R.H. (1997). REAL REVOLUTION; Rob Glaser's RealVideo and his recent deals with MCI and Microsoft promise to transform the Net - and transform his company, Progressive Networks, into a kingmaking Internet Powerhouse. *Wired.* Vol. 5, No. 10, Pp. 122-127.

Robertson, S.E. (1977). The probability ranking principle in IR. *Journal of Documentation.* Vol. 33, No. 4, Pp. 294-304.

Robertson, S.E. (1992). On the evaluation of IR systems. *Information Processing & Management.* Vol. 28, No. 4, Pp. 457-466.

Robertson, S.E., Walker, S., Beaulieu, M.M., Gatford, M., and Payne, A. (1996). Okapi at TREX-4. *The Fourth Text Retrieval Conference (TREC-4)* (ed. D.K. Harman), National Institute of Standards and Technology, Gaithersburg, MD.

Rocchio, J.J., Jr. (1968). Relevance Feedback in Information Retrieval. *The SMART Retrieval System - Experiments in Automatic Document Processing.* (Ed. G. Salton) Prentice-Hall, Englewood Cliffs, N.J.

Rose, D.E. and Belew, R.K. (1991). A connectionist and symbolic hybrid for improving legal research. *International Journal of Man-Machine Studies.* Vol. 35, No. 1, Pp. 1-33.

Rosengren, P. (1994). ER-based information retrieval in a mixed database. *Lecture notes in computer science.* Vol. 823, Pp. 426-437.

Rowe, N.C. (1988). *Artificial Intelligence through Prolog.* Prentice-Hall, London, UK.

Rowley, J. (1994). The controlled versus natural indexing languages debate revisited: a perspective on information retrieval practice and research. *Journal of Information Science.* Vol. 20, No. 2, Pp. 108-119.

Roziç, H. D. and Dimec, J. (1994). Unified Access to Medical Bibliographic Databases. *Proceedings of MIE'94.* (Eds. P. Barahona, M. Veloso, and J. Bryant). Pp. 517-522. Lisbon, Portugal.

Ryle, G. (1949). *The concept of mind.* Hutchinson, London, UK.

Salton, G. (1989). *Automatic Text Processing; The transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley Publishing Company, Reading, MA.

Salton, G. (1991). Developments in Automatic Text Retrieval. *Science.* Vol. 253, No. 8, Pp. 974-980.

Salton, G. and Buckley, C. (1990a). *Approaches to Global Text Analysis.* TR90-1113. Dept. of Computer Science, Cornell University, Ithaca, NY.

Salton, G. and Buckley, C. (1990b). Improving retrieval performance by relevance feedback. *Journal of the Americal Society for Information Science.* Vol. 41, No. 4, Pp. 288-297.

Salton, G., Fox, E.A., and Voorhees, E. (1985). Advanced Feedback Methods in Information Retrieval. *Journal of the American Society for Information Science.* Vol. 36, No. 3, Pp. 200-210.

Salton, G., Fox, E.A., and Wu, H. (1983). Extended Boolean Retrieval. *Communications of the ACM.* Vol. 26, No. 12, Pp. 1022-1036.

Salton, G. and McGill, M.J. (1983). *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, NY.

Sanderson, M. (1994). Word Sense Disambiguation and Information Retrieval. *Procs. of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* (Eds. W. B. Croft and C.J. van Rijsbergen) ACM, New York, NY.

Savoy J. (1994). Searching information in legal hypertext systems. *Artificial Intelligence and Law.* Vol. 2, Pp. 205-232.

Savoy, J. and Desbois, D. (1991). Information retrieval in hypertext systems: an approach using Bayesian networks. *Electronic Publishing.* Vol. 4, No. 2, Pp. 87-108.

Schank, R.C. (1975). *Conceptual information processing.* North Holland Publishing Co., Amsterdam, The Netherlands.

Schank, R.C. and Abelson, R.P. (1977). *Scripts, plans, goals, and understanding.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Scholtes, J.C. (1996). Wie kan zoeken beheerst Internet. *Automatisering Gids.* Vol. 29, No. 16, p. 15. (In Dutch) {The person who understands searching rules the Internet.}

Schuyler, P.L., Hole, W.T., Tuttle, M.S., Sherertz, D.D. (1993). The UMLS Metathesaurus: representing different views of biomedical concepts. *Bull Med Libr Assoc.* Vol. 81, No. 2, Pp. 217-22.

Searle, J.R. (1969). *Speech Acts. An Essay in the Philosophy of Language.* Cambridge University Press, Cambridge, UK.

Searle, J.R. and VanderVeken, D. (1985). *Foundations of Illocutionary Logic.* Cambridge University Press, Cambridge, UK.

Sebastiani, F. (1998). Trends in… A Critical Review; on the Role of Logic in Information Retrieval. *Information Processing and Management.* Vol. 34, No. 1, Pp. 1-18.

Shannon, C.E. and Weaver, W. (1949). *The Mathematical Theory of Communication.* MIT Press, Cambridge, CA.

Shapiro, S.C. (Ed.) (1992). *Encyclopedia of Artificial Intelligence.* Wiley-Interscience publication.

Sheridan, P. and Ballerini, J.P. (1996). Experiments in Multilingual Information Retrieval using the SPIDER system. *Procs. Of the 19th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* (Eds. H.–P. Frei, D. Harman, P. Schäuble, and R. Wilkinson) Pp. 58-65. ACM, New York, NY.

Sheridan, P., Braschler, M., and Schäuble, P. (1997). Cross-language Information Retrieval in a Multilingual Legal Domain. *Lecture notes in computer science.* Vol. 1324, Pp. 253-268. Springer-Verlag, Berlin, Germany.

Shute, S.J. and Smith, P.J. (1993). Knowledge-based search tactics. *Information Processing and Management.* Vol. 29, No. 1. Pp. 29-45.

Shute, S.J., Smith, P.J., Krawczak. D., Chignell, M.H., and Sater, M. (1986). Enhancing cognitive

compatibility in a knowledge-based information retrieval system. In: *1986 Fall Industrial Engineering Conference Proceedings.* Pp. 111-115.

Siau, K.L., Chan, H. C., and Wei, K.K. (1995). Eindgebruikers communiceren met systemen. *Informatie* Vol. 37, No. 12, Pp. 660-668. (In Dutch.) {End-users communicate with systems.}

SKBS B3-project (1991). *Multi-Media Knowledge-Based Systems.* Project proposal SKBS category B. Stichting Knowledge-Based Systems.

Small, S., Cottrell, G., and Tannenhaus, M. (1988). *Lexical Ambiguity Resolution.* Morgan Kaufmann, Palo Alto, CA.

Smeaton, A.F. (1996). An Overview of Information retrieval. *Information Retrieval and Hypertext.* (Eds. M. Agosti and A.F. Smeaton) Pp. 3-25.

Smeaton, A. F. and Crimmins, F. (1997). Relevance Feedback and Query Expansion for Searching the Web: A Model for Searching a Digital Library. *Research and Advanced Technology for Digital Libraries. First European Conference, ECDL '97.* (Eds. C. Peters and C. Thanos) Pp. 99-112. Springer Verlag Lecture Notes in Computer Science, Berlin, Germany.

Smeaton, A.F. and Van Rijsbergen, C.J. (1983). The Retrieval Effects of Query Expansion on a Feedback Document Retrieval System. *Computer Journal.* Vol. 26, No. 3, Pp. 239-246.

Smith, P.J., Shute, S.J., Galdes, D., and Chignell, M.H. (1989). Knowledge-based search tactics for an intelligent intermediary system. *ACM Transactions on Information Systems.* Vol. 7, No. 3, Pp. 247-270.

Smithson, S. (1994). *Information Retrieval Evaluation in Practice: A case study approach.* In: Information Processing & Management. Vol. 30, No. 2, Pp. 205-221.

Spink, A. (1994). Term Relevance and Query Expansion: Relation to Design. *Procs. Of the 17th Annual International ACM-Conference on Research and Development in Information Retrieval.* (Eds. W.B. Croft and C.J. van Rijsbergen) Pp. 81-90. *ACM*, New York, NY.

Spink, A. and Saracevic, T. (1997). Interaction in Information Retrieval: Selection and Effectiveness of Search Terms. *Journal of the American Society for Information Science.* Vol. 48, No. 8, Pp. 741-761.

Srihari, R.K. (1995). Automatic Indexing and Content-Based Retrieval of Captioned Images. *Computer.* Vol. 28, No. 9, Pp. 49-56.

Steketee, H. (1997). Zoek zoek zoek. *NRC Handelsblad.* July 26th, p. 33. (In Dutch) {Search search search.}

Stevenhagen, A. and Van Leeuwen, H.B. (1995). *Octrooien; praktische wenken voor uitvinders en managers.* 3rd revised edition. Kluwer, Deventer, The Netherlands. (In Dutch.) {Patents; practical tips for inventors and managers.}

Stolk, A.L. (1990). *Koudetechniek A1: Koudeopwekking.* Fac. der Werktuigbouwkunde en Maritieme Techniek. Delft University of Technology, Delft, The Netherlands. (In Dutch) {Refrigeration technology A1: the generation of cold.}

Swets, J. (1969). Effectiveness of Information Retrieval Methods. *American Documentation.* Vol. 20, No. 1, Pp. 72-89.

Toulmin, S.E. (1959). *The uses of argument.* Cambridge University Press. Cambridge, MA.

Touretzky, D.S. (1986). *The mathematics of inheritance systems.* Morgan Kaufmann, Los Altos, CA.

Turtle, H.R. and Croft, W.B. (1991). Evaluation of an inference network-based retrieval model. *ACM transactions on Information systems.* Vol. 9, No. 3, Pp. 187-222.

Turtle, H.R. and Croft, W.B. (1992). A comparison of text retrieval models. *The Computer Journal.* Vol. 35, No. 3, Pp. 279-290.

UMLS (1994). *UMLS Knowledge Sources.* 5th Experimental Edition. National Library of Medicine, Bethesda, MD.

Van den Herik, H.J. (1988). *Informatica en het menselijk blikveld.* Inaugural Address. Rijksuniversiteit Limburg, Maastricht, The Netherlands. {Computer Science and the human horizon.}

Van der Pol, R.W. (1996*). Two heuristic rules for query formulation in information retrieval.* Internal

Report SKBS/B3.A/95-06. Universiteit Maastricht, Maastricht, The Netherlands.

Van Dijk, Th. (1994). *The Limits of Patent Protection; Essays on the Economics of Intellectual Property Rights.* Ph.D-thesis. Universiteit Maastricht, Maastricht, The Netherlands.

Van Gent, J. and Kraay, W. (1999). Interpreteren van informatie is moeilijk te automatiseren. *Automatisering Gids.* Vol. 32, No. 14, p 21. (In Dutch) {It is difficult to computerise the interpretation of information.}

Van Heijst, G., Schreiber, Th., and Wielinga, B.J. (1997). Using Explicit Ontologies in KBS Development. *International Journal of Human and Computer Studies.* Vol. 42, No. 2/3, Pp. 183-292.

Van Hyfte, D.M.H., De Clerq, P.A., Tjandra-Maga, T.B., Zitman, F.G., and De Vries-Robbé (1999). Modelling the psychoactive drug selection application domain at the knowledge level. *Proceedings of BNAIC* (Eds. E.O. Postma en M. Gyssens). Pp. 187-194. Universiteit Maastricht, Maastricht, The Netherlands.

Van Rijsbergen, C.J. (1979). *Information Retrieval.* 2nd edition. Butterworth & Co Ltd. London, UK.

VIP (1994). Opmerkelijke tijdwinst door snelle retrieval met Trip. *VIP: vakblad voor image processing.* Vol. 6, No. 4, Pp. 8-10. (In Dutch.) {Remarkable gain of time from fast retrieval by Trip.}

Voorhees, E. (1994). Query expansion using lexical-semantic relations. *Procs. of the 17th annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* (Eds. W.B. Croft and C.J. van Rijsbergen) Pp. 61-69. ACM, New York, NY.

Waller, W.G. and Kraft, D.H. (1979). A mathematical model for a weighted Boolean retrieval system. *Information Processing and Management.* Vol. 15, No. 5, Pp. 235-245.

Webster's New Collegiate Dictionary (1975). G.z.C. Mirriam Co, USA.

Wiesman, F.J. (1998). *Information Retrieval by Graphically Browsing Meta-Information.* Ph.D. Thesis Maastricht University. Phidippides, Cadier en Keer, The Netherlands.

Wiesman, F., Hasman, A. and Meulen, M. van der (1994). Uniform Access to Multiple Sources of Information. In: *Proceedings of MIE'94* (Eds. P. Barahona, M. Veloso, and J. Bryant). Pp. 417-421. Lisbon, Portugal.

Williams, B.C. and Bobrow, D.G. (1991). The Physicist, Chemist and Biologist: A Multi-level View on Knowledge Sharing. Position paper for the Workshop on Shared, Reusable Knowledge Bases. SRKB mailing list. *URL*: http://hpp.stanford.edu/email-archives/srkb.messages/18.html

Winograd, T. (1983). *Language as a cognitive process, Volume I: Syntax.* Addison-Wesley, Reading, MA.

Wittgenstein, L. (1953). *Philosophical Investigations.* Basil Blackwell Ltd., Oxford, UK.

Wong, S.K.M., Ziarko, W., and Wong, P.C.M. (1985). Generalized Vector Space Model in Information Retrieval. Procs. of the 8th Annual International *ACM-SIGIR Conference on Research and Development in Information Retrieval.* Pp. 18-25. ACM, New York, NY.

Woods, W.A. (1975). What's in a link: Foundations for Semantic Networks. In: *Representation and Understanding: Studies in Cognitive science.* (Eds. D.G. Bobrow and A. Collins) Academic Press Inc., New York, NY.

Yokoi, T. (1995). The EDR Electronic Dictionary. *Communications of the ACM.* Vol. 38, No. 11, Pp. 42-44.

Zadeh, L.A. (1965). Fuzzy sets. *Information and Control.* Vol. 8, Pp. 338-353.

Zavrel, J. (1996). Neural Navigation Interfaces for Information Retrieval: Are They More than an appealing Idea? *Artificial Intelligence Review.* Vol. 10, Pp. 477-504.

Ze Wang, J., Wiederhold, G., Firschein, O., Wei, S.X. (1997). Content-based image indexing and searching using Daubechies' wavelets. *Int. Journal on Digital Libraries.* Vol. 1, No. 4, Pp. 311-328.

Zipf, G.H. (1949). *Human Behavior and the Principle of Least Effort; an Introduction to Human Ecology.* Addison-Wesley Publishing Company, Cambridge, MA.

Index

Summary

Knowledge-based Query Formulation in Information Retrieval

Our research resides, in the field of Information Retrieval. It aims at improving the effectiveness of *query-based document-search processes*. These processes comprise four sub-processes, viz. (1) describing of documents (usually indexing), (2) query formulating, (3) matching, and (4) inspecting. In a matching process a query is compared to an index list of a document collection, thereby calculating for each document a relevance value for the query. The query is a formal expression meant to indicate the information need of a user; the indexes are expressions meant to characterise the documents in the collection. Thus, a matching process yields a list of documents predicted to be relevant to the user's information need. After inspection, a new query may be formulated using the information gathered. The latter process is called relevance feedback. The effectiveness of query-based document-search processes is expressed by the notions recall and precision. An effective process has a high recall and a high precision, i.e., it identifies many of the relevant documents in the document collection and it identifies hardly any irrelevant documents.

Query-based document-search processes show an effectiveness (recall and precision) which is far from perfect. One possible cause for the poor performance is that the queries formulated are imperfect. This leads to our main research question, in which we restrict ourselves to initial queries (i.e., queries formulated in which no relevance feedback has been applied):

> *How can, in query-based document-search processes, the effectiveness of the formulation of an initial query be improved by utilising represented knowledge?*

The main research question is worked out in three research questions. Here we restrict ourselves to query-based document-search processes using full-text indexing and a Boolean query language. The first research question aims at studying the formulation of queries by human beings. The second question prompts the development of a representation language (Dipe-R) for domain knowledge that supports initial query-formulation processes. The third question aims at the

development of a knowledge-based technique that supports initial query-formulation processes. The latter technique incorporates a two-step approach (substantiated in a program named Dipe-D) in which first the concepts of an information need are determined, and then the actual query is formulated. Both Dipe-R and Dipe-D are used in an experiment to test Dipe-D's effectiveness. The main conclusion of our research is:

> *The effectiveness of query-based document-search processes, with full-text indexing and Boolean matching, can be improved by the two-step approach to initial query formulation of Dipe-R and Dipe-D.*

Chapter 1 introduces query-based document-search processes, by their concepts and performance measures. It also motivates the relevance of our research, indicating that the processes are important in daily life, that they are not highly effective, and that the latter finding can be partly explained by problems in query formulation.

Chapter 2 characterises basic notions that play a role in discussing document-search processes: communication, information, knowledge, and document. Although – or rather because – these are familiar notions, we believe it to be wise to characterise what *we* mean by them, for the sake of effective communication.

Chapter 3 reviews the literature on Information Retrieval, and formulates the above-mentioned research questions. The review comprises various matching models, methods of indexing, and query formulating. Focusing on query formulating, we find that all existing query languages have keys and most of them also have operators. Known difficulties in query formulation are:

1. that a user cannot express his information need in a proper way, since he cannot
   a. use the appropriate operators in a proper way,
   b. conceive the subjects of the information need,
   c. designate the subjects properly (i.e., in correspondence to the document descriptors), and
2. that a user does not know whether the subjects of the information need are present in the document collection, i.e., he does not know whether the document collection can satisfy the information need (this is called a conceptual gap).

It is known that queries can be improved by reformulation, with (1) domain information, and (2) relevance feedback. We expect that a good initial query will converge rather fast to a high effectiveness of the search process, either with or without query reformulation. We regard it as our challenge to improve the effectiveness of the initial query formulation process, by using represented domain knowledge. Hence we arrived at the above-mentioned research questions.

Chapter 4 treats the first research question, aiming at an improved understanding of query formulation: *How are queries formulated adequately, in a search task that serves the writing of a patent application document?* A theoretical analysis of query formulation yields the insight that two sub-processes occur (possibly intertwined): (1) developing an understanding of the knowledge lack, and (2) expressing the information need. A simulation of patent-search tasks yields the following three (tentative) conclusions: (1) the user's understanding of his knowledge lack (i.e., his information need) changes noticeably, i.e., the development of the understanding of the

knowledge lack plays an essential role, (2) concepts are identified by their characteristics (direct or by association), and (3) queries are formulated according to a fixed procedure.

Chapter 5 treats the second research question: *Is it possible to formulate a representation language for domain knowledge that effectively supports initial query-formulation processes?* Developing Dipe-R assesses the question. Dipe-R's general properties are:

1. the representation of thoughts,
2. a distinction between the storage form and presentation form of represented thoughts. The presentation form is cast in sentences of natural language. This supports the language variety needed in query formulation,
3. a distinction between the content of a represented thought and its source: each represented thought is accompanied by source information, existing of (a) an identifier of the person or document which is the source, (b) the moment in which the thought was obtained, and (c) the manner in which it was obtained (observation, communication, etc.). This distinction recognises the dynamic and subjective nature of thoughts (and of the way of expressing). It is deemed useful for finding a proper interpretation of the sentence, and for assessing the credibility of the thought it expresses.

More specific, Dipe-R has two types of represented thoughts. Both types represent binary relations between concepts. One type also includes quantities. By these two types of thoughts, type hierarchies as well as features of concepts can be represented. Moreover, relationships among concepts, designations, and words are represented. The latter is realised by recognising that designations and words, i.e., symbols, are also concepts. In the same manner as other concepts, the binary relations of the two thought types of Dipe-R thus can relate symbols to concepts (including symbols). Only when a represented thought containing a symbol concept is presented to a user, the symbol concept is treated differently from other concepts (an instance is shown). In addition, Dipe-R contains derivation rules and a (simple) derivation procedure. These allow for elementary derivations, such as the inheritance of features, and the use of transitivity. With this structure of Dipe-R, concepts can be identified by their hierarchical relationships and their features.

Chapter 6 treats the third research question: *Is it possible to formulate a knowledge-based technique that effectively supports initial query-formulation processes?* Developing Dipe-D, including a two-step approach based on the lessons of chapter 4, assesses the question. The first step serves the development of the information need, and the identification of concepts. It comprises an iterative loop of posing questions and obtaining answers with explanations. The basic concept-identification loop consults a knowledge representation expressed in Dipe-R, and has four steps: (1) specification of the information need (in an artificial language), (2) disambiguation (if necessary), (3) solution, i.e., identifying represented concepts that meet the specification, and (4) explanation. An extended loop repeats the basic loop several times, in order to create descriptions of information needs that cannot be expressed in single concepts of the knowledge representation consulted.

The language for specification consists of two main expression types, and a small collection of sub-expression types. The main expressions allow for starting a specification and extending another one, respectively. The sub-expressions specify concepts by type hierarchy, and by (other) relations to other concepts. By combining several sub-expressions, a variety of specifications can

be created. The specification language allows for both a broad description and a narrow description. By using the answers, the user may gather the knowledge needed to develop his information need and to create a proper expression thereof. The narrow descriptions of the specification language then may be used to express precisely an established information need, yielding only the concepts needed.

Dipe-D's second step serves the (automatic) transformation of the collection of (represented) concepts identified into a Boolean query. In its basic version the transformation literally uses the designations (including synonyms) of the concepts, and connects them by the disjunction operator. A variant enhances the recall, as follows: it finds alternative designations of the concept (synonyms and lexical variants), creates a sub-query with the individual words (and word variants, and word stems, excluding stopwords, i.e., words known to have little distinguishing power) of each designation, and combines the sub-queries into a query. A next variant further enhances the recall, by adding quasi-synonyms. Another variant enhances the precision: it reduces the ambiguity of a homonymous designation in a query, by including designations related to the intended meaning and excluding terms related to unintended meaning(s) of the homonymous designation. The transformation process described serves the basic concept-identification loop above. The extended loop is served by an extended transformation process: this is essentially identical, but is repeated for each time the basic loop is repeated, and subsequently the resulting (sub)queries are connected by conjunction operators into a single query.

Chapter 7 reports on the implementation of Dipe-D and on an example knowledge representation in Dipe-R. Both are used in the experiments of chapter 8 for testing Dipe-D. The implementation of Dipe-D is mainly done in the programming environment Delphi 2.0 (using the MsWindows operating system). The derivation part is programmed in PROLOG. The program of Dipe-D has two main windows: one for each step. The example knowledge representation lies in the engineering field of refrigeration. It contains approximately 2,300 represented thoughts, including 600 represented concepts. Despite the finding that many thoughts cannot be expressed in Dipe-R, we found it possible to express a fair amount of domain knowledge in Dipe-R.

Chapter 8 reports on experiments with Dipe-D. Each of Dipe-D's two steps is tested in isolation. The concept-identification step is tested by having six test persons each completing three search tasks. In each task, the test person had to identify concepts using Dipe-D with our example representation in Dipe-R, starting from a written search task. We draw three conclusions:

1. Dipe-D supports the user in developing an information need from a given description by features in natural language into a collection of concepts satisfying the description; this yields a concept precision of 100%, i.e., no irrelevant concepts are identified.
2. Dipe-D supports the user in developing an information need from a description by features in natural language into a collection of concepts satisfying the description with a concept recall of some 70%, i.e., less than 100%. In other words, not all relevant concepts are identified.
3. The causes for the observed low concept recall are (1) difficulties in translating the expression of types and features in the tasks into the types and features in the knowledge representation, and (2) insufficient expressive power of Dipe-D's specification language.

The transformation step was tested by comparing transformation by hand to automated transformation by Dipe-D. Eight test persons each completed three query-formulation tasks, in which

tasks the terms and their variants were given to the test persons. Dipe-D did the same tasks. Next, the queries were matched by a Boolean search system, having a document collection of patent abstracts. We draw the conclusion that the queries created by the transformation step of Dipe-D are on average as effective as those created manually by test persons familiar with Boolean document-search processes. Each of the four conclusions in chapter 8 is only tentative, since the scale of the experiments is small. Statistical significance may be obtained in further experiments, as well as proof that the transformation step by Dipe-D outperforms human beings.

Chapter 9 summarises the three research questions, their approaches and the results and conclusions. It ends with the main conclusion given above.

Samenvatting

Kennisgebaseerd formuleren van zoekvragen in Information Retrieval

Het in dit proefschrift beschreven onderzoek is gelegen in het gebied van *Information Retrieval*. Het onderzoek richt zich op het verbeteren van de effectiviteit van *zoekvraaggebaseerde documentenzoekprocessen*. Dergelijke processen omvatten vier deelprocessen, te weten (1) het beschrijven van documenten (meestal indexeren), (2) het formuleren van een zoekvraag, (3) het vergelijken, en (4) het inspecteren. In een vergelijkingsproces wordt een zoekvraag vergeleken met een indexlijst van een documentenverzameling, waarbij voor elk document een relevantiewaarde wordt berekend bij de zoekvraag. De zoekvraag is een formele uitdrukking, die bedoeld is om de informatiebehoefte van een gebruiker aan te duiden; de indexen zijn uitdrukkingen, die bedoeld zijn om de documenten in de verzameling te karakteriseren. Een vergelijkingsproces levert aldus een lijst van documenten die relevant worden geacht voor de informatiebehoefte van de gebruiker. Na het inspecteren kan een nieuwe zoekvraag worden geformuleerd met behulp van de verkregen informatie. Het laatstgenoemde proces wordt relevantieterugkoppeling genoemd. De effectiviteit van zoekvraaggebaseerde documentenzoekprocessen wordt uitgedrukt in *recall* en *precision*. Een effectief proces heeft een hoge recall en een hoge precision, d.w.z., het identificeert bijna al de relevante documenten in de documentenverzameling, en het identificeert nauwelijks irrelevante documenten.

De effectiviteit (recall en precision) van zoekvraaggebaseerde documentenzoekprocessen is verre van perfect. Een mogelijke oorzaak voor de lage effectiviteit is dat de geformuleerde zoekvragen niet goed zijn. Dit brengt ons op onze hoofdonderzoeksvraag, waarin we onszelf tot initiële zoekvragen beperken (d.w.z. zoekvragen die zijn geformuleerd zonder dat relevantieterugkoppeling is uitgevoerd):

Hoe kan, in zoekvraaggebaseerde documentenzoekprocessen, de effectiviteit van het formuleren van een initiële zoekvraag worden verbeterd door het benutten van gerepresenteerde kennis?

De hoofdonderzoeksvraag wordt in drie onderzoeksvragen uitgewerkt. Hierbij beperken we ons tot zoekvraaggebaseerde documentenzoekprocessen met full-text indexering en een Boole'se zoekvraagtaal. De eerste vraag richt zich op het bestuderen van het formuleren van zoekvragen door mensen. De tweede vraag leidt tot het ontwikkelen van een representatietaal (Dipe-R) voor domeinkennis die formuleringsprocessen van initiële zoekvragen ondersteunt. De derde vraag houdt zich bezig met het ontwikkelen van een kennisgebaseerde techniek die formuleringsprocessen van initiële zoekvragen ondersteunt. Laatstgenoemde techniek bevat een tweestapsbenadering (die belichaamd wordt in een programma met de naam Dipe-D). Deze benadering stelt eerst de concepten uit een informatiebehoefte vast, en formuleert vervolgens de feitelijke zoekvraag. Zowel Dipe-R als Dipe-D zijn gebruikt in een experiment voor het testen van de effectiviteit van Dipe-D. De hoofdconclusie van ons onderzoek luidt:

> *De effectiviteit van zoekvraaggebaseerde documentenzoekprocessen, met full-text indexering en een Boole's vergelijkingsproces, kan worden verbeterd door middel van de tweestapsbenadering voor het formuleren van initiële zoekvragen van Dipe-R en Dipe-D.*

Hoofdstuk 1 introduceert zoekvraaggebaseerde documentenzoekprocessen aan de hand van relevante begrippen en prestatiemaatstaven. Het geeft ook een motivering van ons onderzoek, waarbij aangegeven wordt dat de processen belangrijk zijn in het dagelijks leven, dat ze niet erg effectief zijn, en dat deze laatste bevinding ten dele kan worden verklaard uit de moeilijkheden bij het formuleren van zoekvragen.

Hoofdstuk 2 kenmerkt basisbegrippen die een rol spelen bij het bespreken van documentenzoekprocessen: communicatie, informatie, kennis, en document. Alhoewel – of liever omdat – dit bekende begrippen zijn, achten wij het raadzaam om aan te geven wat *wij* er mee bedoelen, ter bevordering van een effectieve communicatie.

Hoofdstuk 3 bespreekt literatuur over Information Retrieval, en formuleert de hogergenoemde onderzoeksvragen. De bespreking omvat vergelijkingsmodellen, indexeringsmethoden, en het formuleren van zoekvragen. Onze aandacht is in eerste instantie gevestigd op het formuleren van zoekvragen. Het is bekend dat alle bestaande zoekvraagtalen sleutels hebben en de meeste bovendien operatoren. Twee bekende moeilijkheden bij het formuleren van zoekvragen zijn:

1. dat een gebruiker zijn informatiebehoefte niet op de juiste manier kan uitdrukken, aangezien hij niet a. de juiste operatoren op de juiste manier kan gebruiken, b. de onderwerpen van de informatiebehoefte kan bedenken, c. de onderwerpen juist kan aanduiden (d.w.z. in overeenstemming met de documentbeschrijvingen), en
2. dat een gebruiker niet weet of de onderwerpen van de informatiebehoefte aanwezig zijn in de documentenverzameling, oftewel hij weet niet of de documentenverzameling de informatiebehoefte kan vervullen (dit wordt een conceptuele kloof genoemd).

Het is bekend dat zoekvragen kunnen worden verbeterd door herformulering, met (1) domeininformatie, en (2) relevantieterugkoppeling. Wij verwachten dat een goede initiële zoekvraag tamelijk snel naar een hoge effectiviteit van het zoekproces zal convergeren, ongeacht of een herformulering van de zoekvraag plaatsvindt. We beschouwen het als een uitdaging om de effectiviteit van het formuleringsproces van initiële zoekvragen te verbeteren, door gebruik te

maken van gerepresenteerde domeinkennis. De hierboven genoemde onderzoeksvragen zijn hierop gebaseerd.

Hoofdstuk 4 behandelt de eerste onderzoeksvraag, die gericht is op een verbeterd begrip van het formuleren van zoekvragen: *Hoe worden zoekvragen adequaat geformuleerd, in een zoektaak die het schrijven van een octrooiaanvraag dient?* Een theoretische analyse van het formuleren van zoekvragen leidt tot het inzicht dat er twee (mogelijk verweven) deelprocessen optreden: (1) het ontwikkelen van het begrip van het kennistekort (de informatiebehoefte), en (2) het uitdrukken van de informatiebehoefte. Een simulatie van zoektaken naar octrooien leidt tot de volgende drie (voorlopige) conclusies: (1) het begrip van de gebruiker in het eigen kennistekort (zijn informatiebehoefte) wijzigt aanzienlijk, d.w.z. de ontwikkeling van het begrip van het kennistekort speelt een wezenlijke rol, (2) concepten worden geïdentificeerd aan de hand van hun kenmerken (rechtstreeks of via associatie), en (3) zoekvragen worden volgens een vaste procedure geformuleerd.

Hoofdstuk 5 behandelt de tweede onderzoeksvraag: *Is het mogelijk om een representatietaal voor domeinkennis te formuleren die op effectieve wijze initiële formuleringsprocessen ondersteunt?* De aanpak van deze vraag leidt tot de ontwikkeling van Dipe-R. De algemene kenmerken van Dipe-R zijn:

1. de representatie van gedachten,
2. het onderscheiden van de opslagvorm en presentatievorm van gerepresenteerde gedachten. De presentatievorm bestaat uit zinnen in natuurlijke taal. Dit ondersteunt de taalverscheidenheid die nodig is bij het formuleren van zoekvragen,
3. het onderscheiden van de inhoud van een gerepresenteerde gedachte en zijn bron: elke gerepresenteerde gedachte wordt vergezeld van broninformatie, die bestaat uit (a) een aanduiding van de persoon of het document dat de bron is, (b) het moment van verwerving van de gedachte, en (c) de wijze van verwerving (waarneming, communicatie, etc.). Dit onderscheid onderkent de tijdsafhankelijke en subjectieve aard van gedachten (en van de wijze van uitdrukken). Het wordt van belang geacht voor het vinden van een juiste interpretatie van de zin, en voor het inschatten van de geloofwaardigheid van de gedachte die erdoor wordt uitgedrukt.

Meer in het bijzonder heeft Dipe-R twee typen gerepresenteerde gedachten. Beide typen representeren relaties tussen concepten. Een type bevat ook hoeveelheden. Met deze twee typen gedachten is het mogelijk om zowel typehiërarchieën als kenmerken van concepten te representeren. Bovendien worden relaties tussen concepten, aanduidingen, en woorden gerepresenteerd. Het laatste wordt verwezenlijkt door te onderkennen dat aanduidingen en woorden, d.w.z., symbolen, ook concepten zijn. Op dezelfde wijze als andere concepten kunnen de binaire relaties uit de twee gedachtetypen in Dipe-R dus symbolen relateren aan concepten (inclusief symbolen). Pas wanneer een gerepresenteerde gedachte die een symboolconcept bevat aan de gebruiker wordt gepresenteerd wordt het symboolconcept anders behandeld dan andere concepten (er wordt een exemplaar getoond). In aanvulling op het voorgaande bevat Dipe-R afleidingregels en een (eenvoudige) afleidingsprocedure. Deze maken eenvoudige afleidingen mogelijk, zoals het overerven van kenmerken, en het gebruiken van transitiviteit. Met deze structuur van Dipe-R kunnen concepten worden geïdentificeerd aan de hand van hun hiërarchische relaties en hun kenmerken.

Hoofdstuk 6 behandelt de derde onderzoeksvraag: *Is het mogelijk om een kennisgebaseerde techniek te formuleren die op effectieve wijze initiële formuleringsprocessen van zoekvragen ondersteunt?* De ontwikkeling van Dipe-D behandelt deze vraag, met een tweestapsbenadering die gebaseerd is op de lessen uit hoofdstuk 4. De eerste stap dient voor het ontwikkelen van de informatiebehoefte en het identificeren van concepten. Het omvat een iteratieve lus van vragen stellen en antwoorden krijgen met uitleg. De elementaire concept-identificatielus raadpleegt een kennisrepresentatie uitgedrukt in Dipe-R en bevat vier stappen: (1) specificatie van de informatiebehoefte (in een kunstmatige taal), (2) desambiguering (indien nodig), (3) het oplossingsproces, d.w.z. identificeren van gerepresenteerde concepten die voldoen aan de specificatie, en (4) uitleg. Een uitgebreide lus herhaalt de elementaire lus verscheidene keren, teneinde beschrijvingen van informatiebehoeften te creëren die niet kunnen worden uitgedrukt in enkelvoudige concepten uit de geraadpleegde kennisrepresentatie.

De specificatietaal bestaat uit twee hoofdexpressietypen en een kleine verzameling subexpressietypen. De hoofdexpressies maken het mogelijk respectievelijk een specificatie te beginnen en uit te breiden. De subexpressies specificeren concepten aan de hand van een typehiërarchie en door middel van (andere) relaties tot andere concepten. Door meerdere subexpressies te combineren kan een verscheidenheid van specificaties worden gecreëerd. De specificatietaal maakt zowel een ruime als een nauwe beschrijving mogelijk. Door de antwoorden te benutten, kan de gebruiker de kennis vergaren die nodig is om zijn informatiebehoefte te ontwikkelen en om een juiste uitdrukking daarvan te creëren. De nauwe beschrijvingen van de specificatietaal kunnen dan worden gebruikt om een gerealiseerde informatiebehoefte zodanig precies te beschrijven, dat de beschrijving slechts de benodigde concepten oplevert.

De tweede stap van Dipe-D dient voor het (automatisch) transformeren van de verzameling geïdentificeerde (gerepresenteerde) concepten tot een Boole'se zoekvraag. In de basisuitvoering gebruikt de transformatie letterlijk de aanduidingen (inclusief synoniemen) van de concepten en verbindt ze door de disjunctie operator. Een variant bevordert de recall, op de volgende wijze: hij vindt andere aanduidingen van het concept (synoniemen en lexicale varianten), creëert een subzoekvraag met de losse woorden (en woordvarianten, en woordstammen, met uitzondering van stopwoorden, d.w.z. woorden die bekend zijn om hun geringe onderscheidend vermogen) van elke aanduiding, en combineert de subzoekvragen tot een zoekvraag. Een volgende variant bevordert de recall verder, door quasi-synoniemen toe te voegen. Nog een andere variant bevordert de precision: hij verkleint de ambiguïteit van een homonieme aanduiding in een zoekvraag, door in de zoekvraag aanduidingen te noemen die zijn gerelateerd aan de bedoelde betekenis, en aanduidingen uit te sluiten die zijn gerelateerd aan de onbedoelde betekenis(sen) van de homonieme aanduiding. Het beschreven transformatieproces dient voor de bovengenoemde elementaire concept-identificatielus. Voor de uitgebreide lus is een uitgebreid transformatieproces: dit is in hoofdzaak identiek, maar wordt elke keer herhaald dat de elementaire lus wordt herhaald. Vervolgens worden de resulterende (deel)zoekvragen door conjunctie operatoren verbonden tot een enkele zoekvraag.

Hoofdstuk 7 rapporteert de implementatie van Dipe-D en een proefkennisrepresentatie in Dipe-R. Beide worden gebruikt in de experimenten uit hoofdstuk 8 voor het testen van Dipe-D. De implementatie van Dipe-D gebeurt in hoofdzaak in de programmeeromgeving Delphi 2.0 (onder het MsWindows besturingssysteem). Het afleidingsgedeelte is in PROLOG geprogrammeerd.

Het programma van Dipe-D heeft twee hoofdschermen: één voor elke stap. De proefkennisrepresentatie ligt op het technische vakgebied koudetechniek. Hij bevat ongeveer 2.300 gerepresenteerde gedachten, daarbij zijn 600 gerepresenteerde concepten inbegrepen. Ondanks de bevinding dat menig gedachte niet in Dipe-R kan worden uitgedrukt, hebben wij ervaren dat het mogelijk is om een flinke hoeveelheid domeinkennis in Dipe-R uit te drukken.

Hoofdstuk 8 rapporteert experimenten met Dipe-D. Elk van de twee stappen uit Dipe-D is afzonderlijk getest. De concept-identificatiestap is getest door zes personen ieder drie zoekopdrachten te laten uitvoeren. In elke taak moet de testpersoon concepten identificeren met Dipe-D en onze proefrepresentatie in Dipe-R. Er wordt uitgegaan van een geschreven zoekopdracht. We trekken drie conclusies:

1. Dipe-D ondersteunt de gebruiker bij het ontwikkelen van een informatiebehoefte uit een gegeven beschrijving in natuurlijke taal op grond van kenmerken tot een verzameling concepten die elke voldoen aan de beschrijving; dit resulteert in een concept precision van 100%, d.w.z. dat er geen irrelevante concepten worden geïdentificeerd.
2. Dipe-D ondersteunt de gebruiker in het ontwikkelen van een informatiebehoefte uit een gegeven beschrijving in natuurlijke taal op grond van kenmerken tot een verzameling concepten die voldoen aan de beschrijving met een concept recall van ongeveer 70%, d.w.z. minder dan 100%. Anders gezegd, niet alle relevante concepten worden geïdentificeerd.
3. De oorzaken van de waargenomen lage concept recall zijn (1) moeilijkheden bij het vertalen van de uitdrukkingswijze van typen en kenmerken in de zoekopdrachten naar de uitdrukkingswijze van typen en kenmerken in de kennisrepresentatie, en (2) onvoldoende uitdrukkingskracht van de specificatietaal uit Dipe-D.

De transformatiestap werd getest door de handmatige transformatie te vergelijken met de geautomatiseerde transformatie door Dipe-D. Acht proefpersonen voeren elk drie formuleringsopdrachten van zoekvragen uit. In de opdrachten zijn de termen en hun varianten aan de proefpersonen gegeven. Dipe-D voert dezelfde opdrachten uit. Vervolgens zijn de zoekvragen verwerkt door een Boole's zoeksysteem met een documentenverzameling uit octrooi-uittreksels. We concluderen dat de zoekvragen die door de transformatiestap van Dipe-D zijn gecreëerd gemiddeld net zo effectief zijn als de zoekvragen die handmatig zijn gecreëerd door proefpersonen welke bekend zijn met Boolese documentenzoekprocessen. Elk van de vier conclusies in hoofdstuk 8 is slechts voorlopig, aangezien de omvang van de experimenten gering is. Aanvullende experimenten kunnen statistische significantie opleveren, alsmede aantonen dat de tweede stap van Dipe-D het van de mens wint.

Hoofdstuk 9 resumeert de drie onderzoeksvragen, de aanpakken ervan, en de resultaten en conclusies. Het eindigt met de hierboven vermelde hoofdconclusie.

Curriculum Vitae

Ruud van der Pol was born in Haarlem, on the 9th of October 1962. From 1975 to 1981 he attended the St. Michaël College (Atheneum B). From 1981 to 1988 he studied Mechanical Engineering at the Delft University of Technology, where he majored in Refrigeration and Indoor Climate Control (on a simulation model of refrigerant condensers).

In 1989 he did his military service in the Royal Dutch Army, as an officer in civil vocational training. In March 1990 he joined Fokker Space and Systems in Amsterdam, as a thermal engineer. He worked on the design, the simulation and the testing of spacecrafts, focusing on their thermal behaviour. In May 1991 he joined Exterpatent in The Hague (currently EP&C in Rijswijk), where he worked as a (trainee) patent attorney in mechanical engineering.

Being given the opportunity to realise an old wish, he became a Ph.D. student at the Rijksuniversiteit Limburg, in July 1992. (The Rijksuniversiteit was later renamed into Universiteit Maastricht.) There he participated in the ARCHIMEDES project, exploring various subject areas (including knowledge modelling on economic exchange rates). In 1994 he took up the research resulting in this thesis, on Information Retrieval and Knowledge Representation.

Since March 1999 he works as a freelance consultant. His activities are:
– Giving advice in patenting Information Retrieval software, for Eidetica in Amsterdam,
– Designing analysis software for patent portfolios, for EP&C in Rijswijk,
– Developing a task management method and software, digital files and (other) knowledge management software, with/for EP&C in Rijswijk.

Dissertatiereeks

In 2000 zijn de volgende SIKS-dissertaties verschenen.

2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance
Promotor: prof.dr. J.C. van Vliet (VU)
Promotiedatum: 28 maart 2000

2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management
Promotores: prof. dr. P.M.E. De Bra
prof. dr. R.H. McClatchey
Copromotor: dr. P.D.V. van der Stok
Promotiedatum: 29 mei 2000

2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.
Promotor: Prof. dr. B.J. Wielinga
Co-promotor: Dr. P.A.A. van den Besselaar
Promotiedatum: 20 juni 2000

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design
Promotor: Prof. dr. J.C. van Vliet
Co-promotores: Dr. G.C. van der Veer
Dr. M.J. Tauber
Promotiedatum: 10 oktober 2000

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.
Promotores: Prof.dr. H.J. van den Herik (UM/RUL)
Prof.dr.ir. J.L.G. Dietz (TUD)
Prof.dr.ir. A. Hasman (UM)
Promotiedatum: 14 september 2000